Untersuchung und Vergleich UML-basierter Entwicklungswerkzeuge

eingereicht von:

Oliver Grünsteidl

Jochen Lehner

Diplomarbeit

zur Erlangung des akademischen Grades

Magister rerum socialium oeconomicarumque (Mag. rer. soc. oec.)

Sozial- und Wirtschaftswissenschaftliche Fakultät

Universität Wien

Technisch-Naturwissenschaftliche Fakultät

Technische Universität Wien

Studienrichtung: Wirtschaftsinformatik

Begutachter:

a.o. Univ.-Prof. Dr. Martin Hitz

Wien, im Februar 2000

Inhaltsverzeichnis

1 Einleitung 5
1.1 Allfälliges 6
1.2 Aufbau 6
1.3 Testbedingungen 7
1.4 Danksagung 8

2 Übersicht der getesteten Werkzeuge 9
2.1 Cool:Jex 4.1 9
2.2 objectiF 3.2b 12
2.3 Objekttechnologie-Werkbank 2 15
2.4 Paradigm Plus Enterprise 3.6 18
2.5 Rational Rose 98 Enterprise SR-1 20
2.6 SELECT Enterprise 6.0e SR-4 24
2.7 Software through Pictures/UML 7.1 27
2.8 WithClass 99 Enterprise 30
2.9 Zusammenfassung 33

3 UML-Konformität in der Darstellung 35

4 Kriterienkatalog 38
4.1 Methodenbasierte Kriterien 38
4.2 Funktionale Kriterien 41
4.3 Ergonomie 47

5 Evaluationen 52
5.1 Cool:Jex 4.1 52
5.1.1 Methodenbasierte Kriterien 52
5.1.2 Funktionale Kriterien 55
5.1.3 Ergonomie 63
5.2 objectiF 3.2b 67
5.2.1 Methodenbasierte Kriterien 67
5.2.2 Funktionale Kriterien 69
5.2.3 Ergonomie 74
5.3 Objekttechnologie-Werkbank 2 79
5.3.1 Methodenbasierte Kriterien 79
5.3.2 Funktionale Kriterien 82
5.3.3 Ergonomie 87
5.4 Paradigm Plus Enterprise 3.6 92
5.4.1 Methodenbasierte Kriterien 92
5.4.2 Funktionale Kriterien 94
5.4.3 Ergonomie 98
5.5 Rational Rose 98 Enterprise SR-1 101
5.5.1 Methodenbasierte Kriterien 101

5.5.2 Funktionale Kriterien 104
5.5.3 Ergonomie 110
5.6 SELECT Enterprise 6.0e SR-4 114
5.6.1 Methodenbasierte Kriterien 114
5.6.2 Funktionale Kriterien 116
5.6.3 Ergonomie 123
5.7 Software through Pictures/UML 7.1 127
5.7.1 Methodenbasierte Kriterien 127
5.7.2 Funktionale Kriterien 130
5.7.3 Ergonomie 139
5.8 WithClass 99 Enterprise 142
5.8.1 Methodenbasierte Kriterien 142
5.8.2 Funktionale Kriterien 144
5.8.3 Ergonomie 149

6 Empirische Bewertung der Werkzeug-Bedienung 153
6.1 Zugrundeliegendes Projekt - 'Bestellverwaltung' 153
6.1.1 Natürlichsprachliche Spezifikation 153
6.1.2 Use-Case Modell 154
6.2 Beschreibung der Analyse 154
6.2.1 Vorgangsweise 154
6.2.2 Erläuterung der tabellarischen Bewertung 154
6.2.3 Einschränkungen 155
6.3 Erstellung des Klassendiagramms 156
6.3.1 Zusammenfassung der Ergebnisse 156
6.3.2 Interpretation der Ergebnisse 156
6.3.3 Klassendiagramm - Cool:Jex 157
6.3.4 Klassendiagramm - objectiF 162
6.3.5 Klassendiagramm - OTW-2 167
6.3.6 Klassendiagramm - Paradigm Plus 172
6.3.7 Klassendiagramm - Rational Rose 176
6.3.8 Klassendiagramm - SELECT Enterprise 180
6.3.9 Klassendiagramm - StP/UML 184
6.3.10 Klassendiagramm - WithClass 190
6.4 Erstellung des Use-Case- und Sequenzdiagramms 194
6.4.1 Zusammenfassung der Ergebnisse 194
6.4.2 Interpretation der Ergebnisse 194
6.4.3 Use-Case-/Sequenzdiagramm - Cool:Jex 195
6.4.4 Use-Case-/Sequenzdiagramm - objectiF 198
6.4.5 Use-Case-/Sequenzdiagramm - OTW-2 201
6.4.6 Use-Case-/Sequenzdiagramm - Paradigm Plus 204
6.4.7 Use-Case-/Sequenzdiagramm - Rational Rose 208
6.4.8 Use-Case-/Sequenzdiagramm - SELECT Enterprise 211
6.4.9 Use-Case-/Sequenzdiagramm - StP/UML 214
6.4.10 Use-Case-/Sequenzdiagramm - WithClass 217
6.5 Erstellung des Zustandsdiagramms 220
6.5.1 Zusammenfassung der Ergebnisse 220
6.5.2 Interpretation der Ergebnisse 221
6.5.3 Zustandsdiagramm - Cool:Jex 221
6.5.4 Zustandsdiagramm - Paradigm Plus 223
6.5.5 Zustandsdiagramm - Rational Rose 226
6.5.6 Zustandsdiagramm - SELECT Enterprise 228
6.5.7 Zustandsdiagramm - StP/UML 231
6.5.8 Zustandsdiagramm - WithClass 233
6.6 Round-Trip Engineering 235
6.6.1 Round-Trip Engineering - Cool:Jex 237
6.6.2 Round-Trip Engineering - objectiF 240
6.6.3 Round-Trip Engineering - OTW-2 242
6.6.4 Round-Trip Engineering - Paradigm Plus 244
6.6.5 Round-Trip Engineering - Rational Rose 246

6.6.6 Round-Trip Engineering - SELECT Enterprise 249
6.6.7 Round-Trip Engineering - StP/UML 252
6.6.8 Round-Trip Engineering - WithClass. 255
6.6.9 Ergebnisse der empirischen Performanz-Messung 258

A Literaturverzeichnis 260

1 Einleitung

In den vergangenen Jahren hat die Größe des Marktes für Softwareentwicklungswerkzeuge deutlich zugenommen. Dies ist vor allem auf die steigende Komplexität heutiger Projekte und die damit verbundene Notwendigkeit des Einsatzes von computerunterstütztem Softwareengineering (*CASE - Computer Aided Software Engineering*) zurückzuführen.

Zuallererst stellt sich die Frage, inwieweit moderne Tools den Entwicklungsprozeß eines Softwareprojekts unterstützen können. Folgende Möglichkeiten sind hierbei von Bedeutung:

Aufgaben von CASE-Tools im Softwareentwicklungsprozeß

- ❑ Graphische Modellierung anhand einer bestimmten, standardisierten Notation
- ❑ Wiederverwendung von Komponenten über mehrere Projekte hinweg
- ❑ Unterstützung und Koordination bei großen und verteilten Projekten (mehrere Entwickler, unterschiedliche Umgebungen etc.)
- ❑ Automatisiertes Generieren von Quellcode in einer gewählten Programmiersprache aus den Informationen des entsprechenden Modells
- ❑ Bewahrung der Konsistenz zwischen Modell und Code (bzw. automatische Synchronisation bei Änderungen)
- ❑ Erstellung verschiedener Arten von Dokumentationen des Projektfortschritts (möglichst in gängigen Textformaten)
- ❑ Sonstige Funktionen, wie Berechnung von Statistiken, Unterstützung des Testvorganges auf Modellebene etc.

Hierdurch ist unschwer zu erkennen, daß die Aufgaben eines effizienten Werkzeugs weit über reine Funktionen zur Erstellung einer graphischen Repräsentation des Softwaresystems hinausgehen. Ziel dieser Arbeit war es nun, eine Auswahl der zur Zeit am Markt befindlichen Entwicklungswerkzeuge zu testen und gegenüberzustellen.

Unterstützung der UML vs. herstellerspezifische Paradigmen

Besonderes Augenmerk wurde dabei auf die Unterstützung der *UML - Unified Modeling Language* gelegt. Seit der Einführung der UML als Standard für objektorientierte Modellierung und des damit verbundenen Bedarfs an entsprechenden Werkzeugen, kommt kaum ein CASE-Tool mehr ohne das Siegel "hundertprozentige UML-Unterstützung" aus. Jedoch liegt hier oft die Problematik im Detail. Viele Hersteller definieren eigene, spezielle Ansätze und Paradigmen für den Softwareentwicklungsprozeß, wie etwa der *Rational Unified Process*, die *SELECT Perspective* oder *Platinum Enterprise Component Modeling*. Da die UML keine Modellierungsmethode bzw. keine bestimmte Vorgehensweise im Softwareentwicklungsprozeß formuliert, werden meist nur einige ausgewählte Konstrukte der UML umgesetzt und mit vorhandenen Methoden vermischt. So ist bei genauerer Betrachtung der Produkte zu erkennen, daß kein Werkzeug wirklich sämtliche Vorgaben der UML exakt umsetzt.

Literatur zu UML

Aufgrund der Zielsetzung dieser Arbeit soll auf eine einführende Erläuterung der UML verzichtet werden. Die Autoren verweisen hierbei auf die große Anzahl verfügbarer Literatur (etwa [Hitz99], [Rumb99] oder [Oest97]), die diese Thematik ausführlich behandelt.

1.1 Allfälliges

- Die hier verwendeten Hardware- und Softwarebezeichnungen sind größtenteils eingetragene Warenzeichen. Es gelten die entsprechenden Schutzbestimmungen.
- Bei der Erstellung dieser Arbeit wurde mit größter Sorgfalt vorgegangen. Die Autoren können jedoch für etwaige fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.
- Für Anmerkungen und Vorschläge können die Autoren gerne unter folgenden e-mail Adressen kontaktiert werden: *Kontakt*
  - Oliver Grünsteidl - oliverg@gmx.at
  - Jochen Lehner - JochenLehner@gmx.at

1.2 Aufbau

Die vorliegende Arbeit gliedert sich in die folgenden wesentlichen Abschnitte:

- In Kapitel 2 erfolgt eine Zusammenfassung und Gegenüberstellung der wichtigsten Funktionen, sowie eine Übersicht der auffallenden Stärken und Schwächen der einzelnen Werkzeuge
- Kapitel 3 widmet sich der UML-konformen graphischen Darstellung ausgewählter Konstrukte
- Kapitel 4 beschreibt den Kriterienkatalog, anhand dessen die Bewertung der Tools vorgenommen wurde
- In Kapitel 5 werden die Ergebnisse der Bewertung der Tools im Detail vorgestellt
- Abschließend wurde die Bedienung der Werkzeuge und die damit verbundene Performanz beim Erstellen eines konkreten Softwareprojekts einer empirischen Analyse unterzogen. Die Resultate finden sich in Kapitel 6.

1.3 Testbedingungen

Zur Relativierung und zum besseren Nachvollziehen der Testergebnisse sollten die folgenden Einschränkungen in Betracht gezogen werden:

- Zum Einsatz kamen zwei Rechnersysteme:
  Konfiguration 1: P2-350, 128 Mb RAM
  Konfiguration 2: Celeron-400, 64Mb RAM
  Betriebsystem jeweils Windows-NT 4.0, SP-6

  Rechnerkonfiguration

- Von einigen Produkten wurden von den Herstellern nur Versionen, die in ihrem Funktionsumfang eingeschränkt waren, zur Verfügung gestellt. Dies bezog sich etwa auf fehlende Möglichkeiten des Mehrbenutzerbetriebs o.ä. Dadurch waren die Autoren aussschließlich auf die (machmal leider nur spärlich vorhandene) Dokumentation der Werkzeuge zur Bewertung einiger Kriterien angewiesen.

  Einschränkungen der Testversionen

- Die einzelnen Tools wurden aufgrund der vorhandenen Möglichkeiten anhand kleiner Projekte getestet, sodaß eine Aussage über die Leistung in großen, verteilten Entwicklungsumgebungen oft auf eher theoretischen und subjektiven Einschätzungen beruht.

  Testumgebung

- Die Funktionen vieler Werkzeuge können mittels umfangreicher Scripts an die entsprechenden Anforderungen angepaßt werden. Im Rahmen dieser Arbeit wurden die Test jedoch auf Basis der Standardeinstellungen durchgeführt. Diese Einschränkung ist aus folgenden Gründen nachvollziehbar:

  Standardeinstellungen, Komplexität von Anpassungen

  1. Die Syntax der, von den Tools verwendeten Scripts ist oftmals von hoher Komplexität, sodaß eine umfassende Behandlung den Rahmen dieser Arbeit gesprengt hätte.
  2. Die Autoren sind der Auffassung, daß ein modernes Tool Möglichkeiten bieten sollte, wichtige und sich häufig ändernde Einstellungen, z.B. bei der Codegenerierung innerhalb der entsprechenden Benutzeroberfläche (etwa über Menüs etc.) zu treffen, sodaß eine Überarbeitung der Scripts beim Einsatz des entsprechenden Werkzeuges nur in speziellen Fällen notwendig ist.

- Leider kann eine Evaluierung von Softwareprodukten nur eine Momentaufnahme der aktuellen Situation widerspiegeln. Da manche Werkzeuge aufgrund der Dauer der Arbeit zeitlich früher getestet wurden als andere, konnte die Funktionalität eventuell bereits vorhandener verbesserter Versionen nicht mehr in die Auswertung aufgenommen werden.

  Bewertung ist eine Momentaufnahme

- Die von der OMG standardisierte Fassung der UML liegt derzeit in der Version 1.3 vor [OMG99]. Da jedoch keines der hier getesteten Werkzeuge die neuen Konstrukte dieser Version unterstützt, bezieht sich die Auswertung auf die UML-Version 1.1. Eine Anzahl von Herstellern hat jedoch bereits eine derartige Unterstützung für zukünftige Versionen der entsprechenden Produkte angekündigt.

  Version der UML

1.4 Danksagung

An dieser Stelle möchten die Autoren vor allem den Mitarbeitern der einzelnen Herstellerfirmen pauschal für deren Unterstützung danken. Besonderer Dank gilt auch dem Betreuer dieser Arbeit, a.o. Univ. Prof. Dr. Martin Hitz.

2 Übersicht der getesteten Werkzeuge

Bevor die Ergebnisse der Auswertung detailliert beschrieben werden, sollen in diesem Kapitel die wichtigsten Eigenschaften und Funktionen der Programme zusammenfassend dargestellt werden. Zusätzlich werden wesentliche positive und negative Merkmale der Produkte angeführt. Am Ende des Kapitels erfolgt eine gegenüberstellende Bewertung anhand einiger ausgewählter Punkte und eine Empfehlung für den Einsatzbereich der einzelnen Tools.

Folgende UML-basierten Softwareentwicklungswerkzeuge wurden analysiert und bewertet:

Werkzeuge

- *Cool:Jex 4.1*
- *objectiF 3.2b*
- *Objekttechnologie-Werkbank 2*
- *Paradigm Plus 3.6*
- *Rational Rose 98 Enterprise SR-1*
- *SELECT Enterprise 6.0e SR-4*
- *Software through Pictures/UML 7.1*
- *WithClass 99 Enterprise*

2.1 Cool:Jex 4.1

Hersteller	Sterling Software, Texas, USA
Webseite	http://www.cool.sterling.com/products/jex
Betriebssysteme	Windows 9x oder Windows NT

Positiv:

- Umfangreiche Funktionalität
- Hochgradig an individuelle Bedürfnisse anpaßbar
- Guter Diagrammeditor
- Anbindung an gängige Textverarbeitungssysteme zur Dokumentation
- Professionelle Mehrbenutzer-Unterstützung

Negativ:

- Benutzerschnittstelle nicht intuitiv
- Für kleine Projekte zu komplex

Das CASE-Werkzeug Cool:Jex basiert auf dem Programm ObjectTeam der Firma Cayenne Software aus Massachusetts. Bei der zur Verfügung stehenden Version des Programmes war der Titel *Cool:Jex* ausschließlich auf dem CD-Label zu finden. Im Programm selbst und in der enthaltenen Online-Dokumentation wurde das Werkzeug als *ObjectTeam 8.1.2* bezeichnet.

Cool:Jex alias ObjectTeam

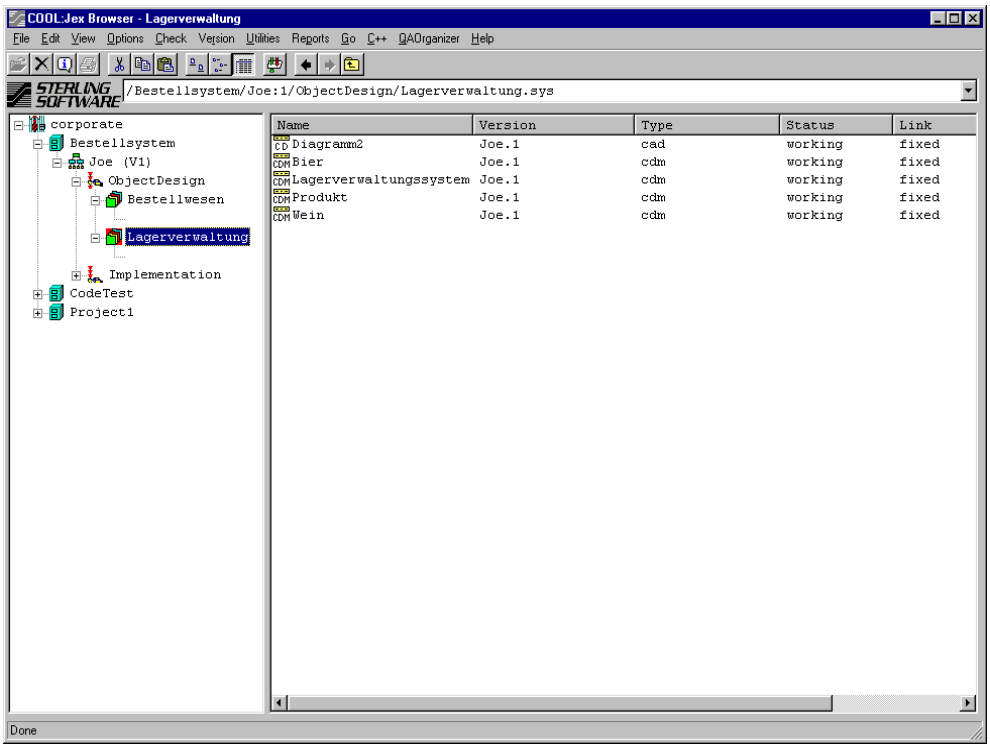

Abb. 2–1
Das Browserfenster von Cool:Jex

Das Programm besitzt eine Client/Server-Architektur, wobei ein Server im Netzwerk das Repository verwaltet. Klienten können selbständig Leseprozesse ausführen, wie Diagramme darstellen, Reports oder Code generieren usw. Kritische Prozesse, die das System verändern müssen allerdings vom Server angefordert werden, wobei dieser alle Aktionen überwacht und die Konsistenz des Systems sicherstellt. Weiters existiert eine Version des Programms, die für Einzelinstallationen gedacht ist (*Standalone*).

Projektdaten werden im sogenannten *Cayenne Repository* gespeichert. Es können mehrere Repositorys angelegt werden, die Herstellerfirma empfiehlt jedoch *ein* Repository für alle Projekte um einen Austausch von Modellierungsinformationen zwischen diesen zu ermöglichen. Das Repository speichert die Daten des Systems teilweise in einer Datenbank und teilweise im Dateisystem. Die eingesetzte Datenbank hängt vom Betriebssystem ab, unter dem Cool:Jex installiert ist. Unter Windows NT und Windows 9x kommt SQL Anywhere zum Einsatz, auf UNIX-Plattformen entweder Informix oder Oracle.

Unterstützung von OMT

Das Werkzeug unterstützt neben der UML auch die OMT-Notation. Wird der OMT-Modus aktiviert, fallen die Diagramme Aktivitäts-, Kollaborations-, Komponenten- und Verteilungsdiagramm weg, während *Class Communication Diagram*, *Data Flow Diagram* und *Message Generalization Diagram* zusätzlich eingeführt werden.

Wird das Programm gestartet, befindet man sich im Repository-Browser von Cool:Jex, der als geteiltes Fenster mit Menü- und Werkzeugleiste ausgeführt ist. Im linken Teil des Fensters wird der Inhalt des Repositorys mittels

tree controls dargestellt, während im rechten Teil zu ausgewählten Elementen Informationen angezeigt werden.

Struktur eines Projekts

Die erste Ebene der Struktur eines Repositorys bilden *Projekte*. Ein Projekt umfaßt eine oder mehrere Konfigurationsversionen, welche ihrerseits Phasenversionen enthalten. Eine Konfigurationsversion kann aus den Phasen *Analyse*, *Systemdesign*, *Objektdesign* und *Implementation* bestehen. Phasen können nun sogenannte *Systemversionen* beinhalten. Eine Systemversion wird als Paket dargestellt und enthält Modellelemente und Diagramme. Das Werkzeug ist so aufgebaut, daß ein System (Systemversion) in der Analysephase erstellt wird, alle Phasen bis zur Implementationsphase durchläuft, und dort schließlich der Code generiert wird (C++, angeblich u.a. auch Java, Smalltalk, Delphi, CORBA/IDL, Visual Basic, Ada und Eiffel). Um bei diesem Vorgang die Übersicht zu behalten, stellt ObjectTeam eine Funktion zur Verfügung, mit der Unterschiede zwischen Systemversionen zweier Phasen festgestellt und angezeigt werden können.

Codegenerierung in mehreren Zielsprachen

Bedienung nicht intuitiv

Das User Interface des Programms kann leider nicht als intuitiv zu bedienen bezeichnet werden (soweit so komplexe Software überhaupt intuitiv zu bedienen sein kann). Am stärksten fällt auf, daß in den Kontextmenüs vieler Elemente wichtige Befehle fehlen. So muß zum Beispiel zum Anlegen einer neuen Konfigurationsversion in einem Projekt zuerst das Projekt im Repository-Browser markiert werden, und dann über das Hauptmenü (*File - New - Configuration Version*) eine neue Konfigurationsversion angelegt werden. Das Kontextmenü zu dem Projekt enthält zwar einige spezifische Einträge wie *Info*, *Edit Properties* usw., eine neue Konfigurationsversion kann auf diesem Weg jedoch nicht angelegt werden.

Der Diagrammeditor des Programms (siehe Abb. 2–2) ist sehr umfangreich ausgestattet. Er erlaubt die Modifikation von Farben und Zeichensätzen und stellt zusätzlich Werkzeuge zur Verfügung, um Diagramme auf Konsistenz zu überprüfen und Reports über deren Inhalt auszugeben. Weiters ist ein Klassenbrowser verfügbar, der zu den vorhandenen Klassen Informationen wie Generalisierungen und Assoziationen anzeigt.

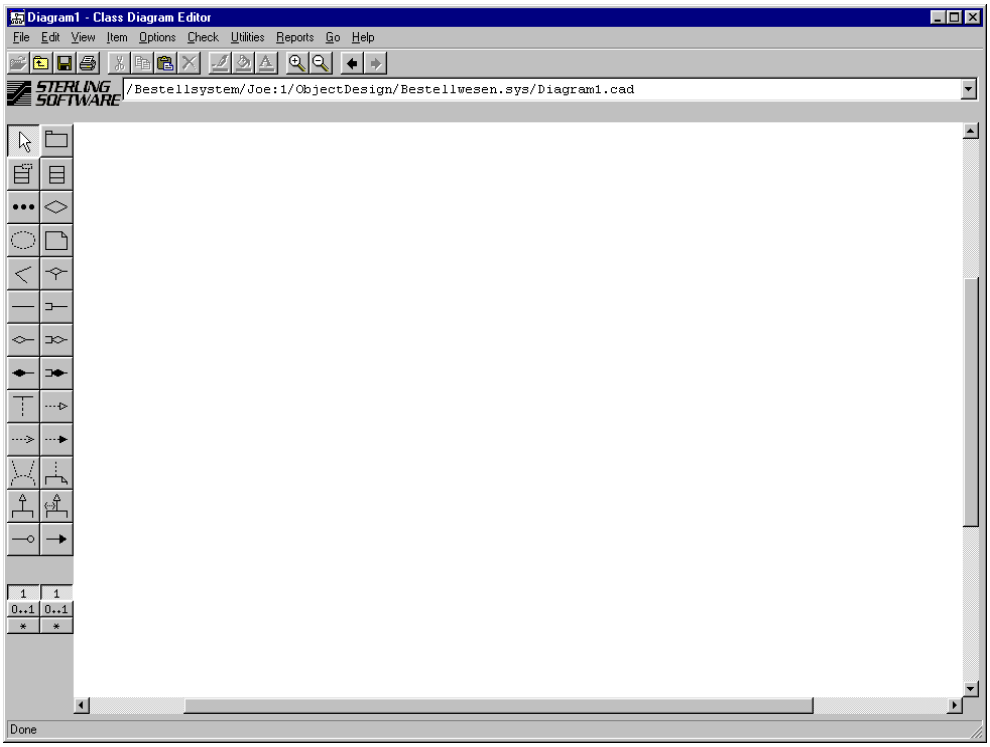

Abb. 2–2
Der Diagrammeditor von Cool:Jex

Reports und Konsistenzprüfungen

Das Programm gibt auf Wunsch Reports über den Inhalt beliebiger Teilaspekte eines Projektes in einem Statusfenster aus. Diese Reports können im Programm ausgedruckt oder in einer Textdatei gespeichert werden.

Es existieren mehrere Prüffunktionen, die bestimmte Aspekte des Projektes auf Fehler untersuchen und das Ergebnis dieser Überprüfung in einem Statusfenster ausgeben. Es besteht die Möglichkeit nur ein einzelnes Diagramm zu überprüfen oder - orthogonal dazu - ausgewählte Modellierungselemente genauer zu untersuchen, wobei Informationen aus allen Diagrammen, in denen diese Elemente vorkommen verwendet werden.

Um Dokumentationen zu bestehenden Projekten automatisch generieren zu können, beinhaltet Cool:Jex sogenannte *DocWriter*-Module, die mit gängigen Textverarbeitungssystemen zusammenarbeiten. Es werden Module für MS-Word unter Windows sowie FrameMaker und Interleaf unter UNIX mitgeliefert.

Anpassung an individuelle Bedürfnisse

Sowohl das Interface, als auch die Funktionalität von Cool:Jex kann an die individuellen Wünsche des Benutzers angepaßt werden um die Produktivität bei der Arbeit mit dem Werkzeug zu erhöhen. Dies geschieht über Umgebungsvariablen (sogenannte *M4-Variablen*), durch Aktivieren bzw. Deaktivieren verschiedener Module und durch Verändern von *customization files*. Zusätzlich bietet Cool:Jex Tastaturkürzeln für die gängigsten Befehle. Diese sind sowohl im Repository-Browser (Hauptfenster) als auch im Diagrammeditor verfügbar.

2.2 objectiF 3.2b

Hersteller	MicroTOOL GmbH, Berlin, Deutschland
Webseite	http://www.microtool.de/objectiF
Betriebssysteme	Windows 9x oder Windows NT

Positiv:

- ❑ Einfache Bedienung
- ❑ Modernes und praktisches GUI
- ❑ Repository basiert auf einem OODBMS
- ❑ Online-Dokumentationen im HTML-Format erzeugbar
- ❑ Code kann schon während der Modellierung händisch ergänzt werden
- ❑ Auch externer Code kann gescannt werden
- ❑ Diagrammausgabe grafisch ansprechend umgesetzt
- ❑ Neben den Windows-Hilfedateien wird eine ausführliche Dokumentation im PDF-Format mitgeliefert
- ❑ Im Mehrbenutzerbetrieb findet Locking auf Objektebene statt
- ❑ Gute Performanz

Negativ:

- ❑ Funktionsumfang eher gering
- ❑ Verteilungs-, Kollaborations- und Aktivitätsdiagramm nicht unterstützt
- ❑ Zustandsdiagramme nur rudimentär umgesetzt
- ❑ Kein Export von Daten aus dem Repository möglich

- Fehlende Tastaturkürzel
- Relativ wenig Einstellungsmöglichkeiten, mit denen sich der Detaillierungsgrad der Diagrammausgabe beeinflussen läßt
- Keine *Undo*-Funktion im Grafikeditor
- Keine Suchfunktionen
- Programm unterstützt keine Versionskontrolle

Geringe Größe des Werkzeugs

objectiF ist im Vergleich zu den meisten seiner Mitbewerber ein eher einfaches Werkzeug. Das installierte Programm belegt inklusive Dokumentation nur knappe 18 Mb auf der Festplatte. In Bezug auf Funktionsumfang und gestalterische Möglichkeiten kann es nicht mit den großen CASE-Werkzeugen mithalten. Dennoch ist objectiF ein leicht zu bedienendes Programm, das bei Projekten, die mit seinem eingeschränkten Funktionsumfang das Auslangen finden, gute Dienste leisten kann.

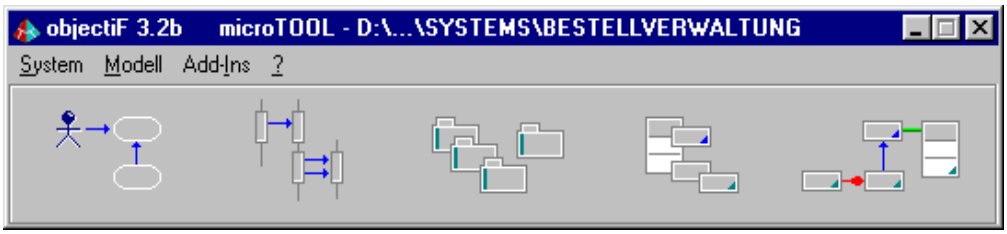

Abb. 2–3
Das Hauptfenster von objectiF

Gelungene Benutzerschnittstelle

Daß das Programm nicht so funktionsüberladen ist wie viele seiner Konkurrenten, erleichtert seine Bedienung (gerade für neue Benutzer) erheblich. Auch sind die Benutzerschnittstelle des Werkzeugs und die Diagrammdarstellung sehr liebevoll umgesetzt. Manchmal wirkt die grafische Darstellung fast zu verspielt, zum Beispiel die *Lifeline* der Objekte im Sequenzdiagramm, die objectiF farbig schattiert anzeigt, um einen 3D-Effekt zu erzeugen. Das Programm, das einen sehr modernen Eindruck macht, stellt die Informationen über das aktuelle Projekt in frei auf dem Desktop plazierbaren Fenstern dar. Das Hauptfenster (siehe Abb. 2–3) ist sehr klein und besitzt nur eine Menü- und eine Werkzeugleiste. Fast die gesamte Programmsteuerung wird durch *tree controls* und Kontextmenüs realisiert. Insgesamt können das GUI und die grafische Darstellung von objectiF als gelungen bezeichnet werden.

Der grafische Editor des Werkzeugs erlaubt es nicht, freie Formen zu zeichnen. Dies wäre um so wichtiger, als objectiF einige UML-Konstrukte auf anderem Wege nicht darstellen kann. Konstrukte, die dargestellt werden, entsprechen dem UML-Standard.

Die Diagrammausgabe auf einem Drucker erfolgt problemlos. Das Werkzeug bietet die Möglichkeit, den Ausdruck zu skalieren, die Detailgenauigkeit in gewissem Rahmen zu verändern und den Ausdruck auf mehrere Druckseiten zu verteilen. In letzterem Fall kann objectiF am Ausdruck Falt- und Kleberänder generieren.

Kaum Bereitstellung von Tastaturkürzeln

Bei der Bedienung des Werkzeugs fällt auf, daß es kaum Tastaturkürzel bereitstellt. Besonders in den Browser-Ansichten, die an den Windows-Dateimanager erinnern, kommt es schon einmal vor, daß man vergeblich auf die Entfernen-Taste drückt, bevor man das Element dann über das Kontextmenü löscht.

Die Daten von Modellen speichert objectiF in einem Repository, das auf einem objektorientierten Datenbanksystem basiert. Es existieren einige Browser (in Form von *tree controls,* siehe Abb. 2–5), mit denen das Repository eingesehen werden kann bzw. Elemente darin bearbeitet werden kön-

nen. Das Programm erlaubt Mehrbenutzerbetrieb, wobei Locking auf Objektebene stattfindet.

Abb. 2–4
Der Diagrammeditor von objectiF

Laufende Konsistenzprüfung

Um den Datenbestand konsistent zu halten, geht objectiF nicht den Weg einer explizit aufzurufenden Konsistenzprüfung, sondern sorgt dafür, daß Inkonsistenzen gar nicht erst entstehen. Dies wird dadurch erreicht, daß Dialoge erst dann vorwärts (*OK*) verlassen werden können, wenn die darin angegebenen Parameter das bestehende Modell in einem konsistenten Zustand belassen.

Dokumentationen zu einem Projekt erstellt objectiF, indem es über COM mit MS-Word kommuniziert. Der Benutzer kann festlegen, welche Elemente und Aspekte des Modells in das Dokument aufgenommen werden und wie detailliert dieses ausfallen soll. Weiters bietet das Werkzeug die Möglichkeit, *online*-Dokumentationen im HTML-Format zu erstellen, die dann beispielsweise über ein Intranet anderen Mitarbeitern zugänglich gemacht werden können. Diese HTML-Dokumente sind gut gegliedert und optisch ansprechend aufbereitet (grafische Schaltflächen zum Navigieren, Informationen auf viele Unterseiten verteilt).

Codegenerierung für C++, Java und SQL

Der Code-Generator von objectiF erstellt standardmäßig C++- oder Java-Code sowie DDL-Dateien in SQL. Dies geschieht über sogenannte *code scripts*, die auch selbst erstellt werden können, um den Generator an jede beliebige Sprache anzupassen. Im Programm kann der Benutzer zu jeder Klasse ein Texteditorfenster öffnen, in dem der zugehörige Code angezeigt wird. Hier kann auch händisch Code eingefügt werden, der dann in die generierten Quellcodedateien übertragen wird.

objectiF ist in der Lage, extern erzeugten Code einzulesen und zu interpretieren. Dabei ist jedoch zu beachten, daß sämtliche Programmzeilen, die über die Deklaration hinausgehen, in eigens dafür vorgesehenen, durch spezielle Kommentare gekennzeichneten Blöcken stehen müssen. Ist dies nicht der Fall, wird der Code ignoriert und geht bei einer folgenden Vorwärtsgenerierung verloren.

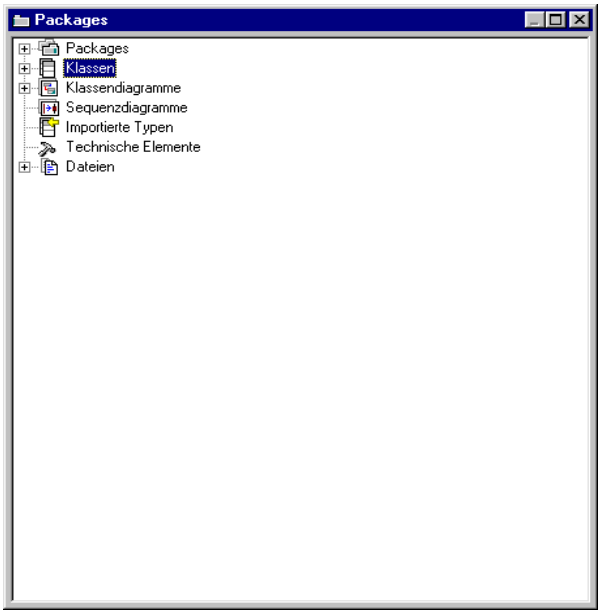

Abb. 2–5
Das Browserfenster von objectiF

objectiF arbeitet mit der Prozeßmanagementsoftware In-Step von Micro-TOOL zusammen. Mit diesem Programm lassen sich aus einem objectiF-Modell bestimmte Aktivitäten ableiten, und es ist möglich, diesen Termine und Ressourcen zuzuordnen.

2.3 Objekttechnologie-Werkbank 2

Hersteller	OWiS Software GmbH, Illmenau, Deutschland
Webseite	http://www.otw.de/
Betriebssysteme	Windows 9x oder Windows NT

Positiv:

- ❑ Typisches *Windows Look & Feel* mit Standard-Menüstruktur und gewohnten Steuerelementen
- ❑ Repository basiert auf einem OODBMS
- ❑ Sequenzdiagramm gut umgesetzt
- ❑ Umfangreiche Einstellungsmöglichkeiten für die grafische Anzeige
- ❑ Einfache Navigation im Systemmodell über ein Kontextmenü
- ❑ Gute Reportausgabe
- ❑ *Round-Trip Engineering* möglich
- ❑ Gut lesbare und umfangreiche Dokumentation in Form mehrerer MS-Word-Dateien, die zugleich als kontextsensitive Hilfe fungieren
- ❑ Gute Performanz

Negativ:

- ❑ Zustands- und Aktionsdiagramm nicht unterstützt
- ❑ Kaum statistische Angaben über den Inhalt des Repositorys

- ❑ Im Mehrbenutzerbetrieb ist ein Sperren im Master-Repository nur auf Paketebene möglich
- ❑ Einige UML-Konstrukte fehlen
- ❑ Keine Import- oder Exportfunktionen
- ❑ Ausdruck von großen Diagrammen unzulänglich
- ❑ Wenig Möglichkeiten, die Produktivität zu erhöhen (Tastaturkürzel, Automation von oft benötigten Arbeitsschritten)
- ❑ Hilfedatei als MS-Word Dokument mit Sprungmarken, keine so komfortable Suche möglich wie in herkömmlichen Hilfedateien

OTW-2 geht in mancher Hinsicht andere Wege als seine Mitbewerber. Das Zustandsdiagramm wurde beispielsweise nicht implementiert, stattdessen wurde das sogenannte *Objekt-Prozeß-Modell (OPM)* in das Programm aufgenommen. Dieses ist eine in [Burk94] definierte Notation für die objektorientierte Prozeßmodellierung. Sie ermöglicht die Modellierung von Vor- und Nachbedingungen. OPM kann als Ergänzung zum Sequenzdiagramm und als Alternative zum Zustandsdiagramm gesehen werden.

Modellierung der Systemdynamik basiert auf OPM

Das Programm hält sich bei den vorhandenen Notationselementen ganz gut an den UML-Standard, jedoch muß angemerkt werden, daß einige Konstrukte, die in UML vorgesehen sind, mit dem Werkzeug nicht umzusetzen sind. Hierzu zählen zum Beispiel n-äre und qualifizierte Assoziationen, baumartige Darstellung bei Generalisierungshierarchien (*shared target style*) u.s.w. In diesem Zusammenhang macht sich der Umstand negativ bemerkbar, daß der Diagrammeditor von OTW-2 keine Möglichkeit bietet, freie Formen und Linien zu erstellen. Mit solchen Hilfsmitteln könnte der Benutzer notfalls die fehlenden Notationselemente zumindest graphisch anlegen.

UML wird nur unvollständig abgedeckt

Die Ausgabe von Diagrammen auf einem Drucker ist unzufriedenstellend, sobald diese eine gewisse Größe erreichen. Es ist leider unmöglich, den Ausdruck über mehrere Seiten zu verteilen, wie dies bei fast jedem anderen untersuchten Programm möglich war. In solchen Fällen skaliert OTW-2 die zu druckenden Diagramme auf Blattgröße, was dazu führen kann, daß Beschriftungen kaum noch lesbar sind.

Druckausgabe unbefriedigend

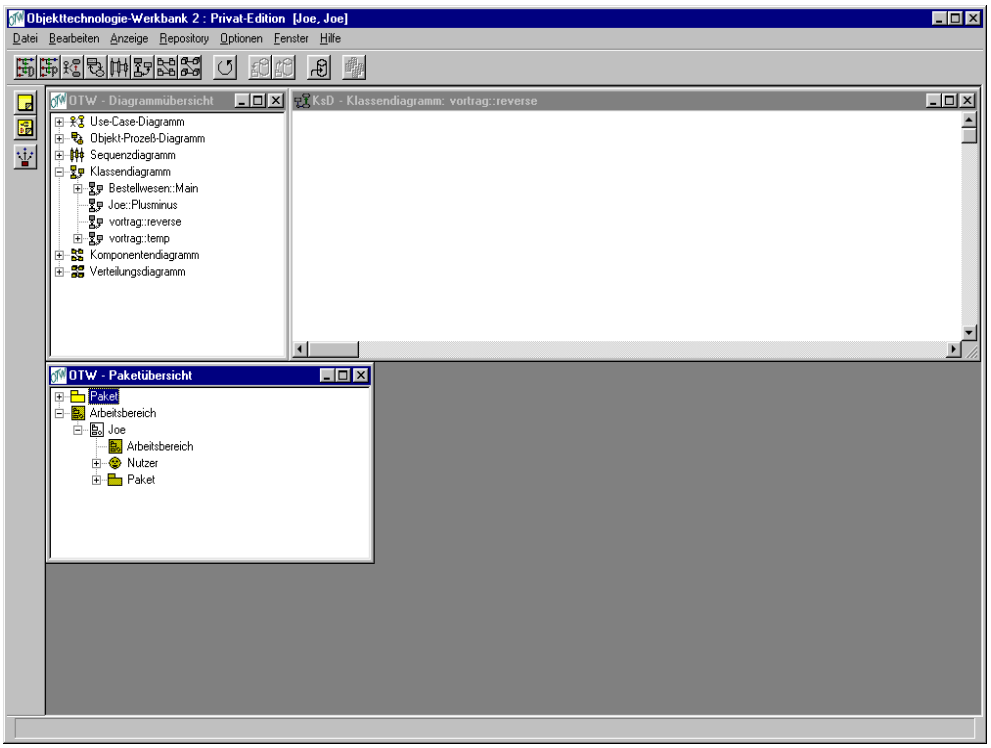

Abb. 2–6
Die Benutzeroberfläche von OTW-2

Erzeugung von Dokumentation

Die Erzeugung von Textdokumenten geschieht mit OTW-2 in zwei Schritten. Zuerst wird eine ASCII-Textdatei mit variablem Inhalt und Genauigkeit erzeugt und speichert. Hierin werden auch Dateinamen und Pfadangaben für die vom Programm gespeicherten Diagrammgrafiken abgelegt, damit diese, wenn gewünscht, in die Dokumentation übernommen werden können. Die Textdatei kann jetzt mit einem beliebigen Textverarbeitungsprogramm mit geeignetem Automationsmechanismus (Makro-, Skriptsprache) weiterverarbeitet werden. Zu OTW-2 wird ein MS-Word Makro mitgeliefert, das umfangreiche und optisch ansprechende Ergebnisse liefert.

Die Daten werden in der OTW-2 in einem zentralen Repository gespeichert, wobei jedes Projekt ein separates Repository erhält. Die Repositorys basieren auf einem objektorientierten Datenbank-Management-System (OODBMS). Die Konsistenz der Modelle wird sichergestellt, indem sofort beim Anlegen von Modellelementen deren Richtigkeit überprüft wird und die Aktion gegebenenfalls mit einem Fehler abgebrochen wird.

Mehrbenutzerbetrieb

Das Programm gestattet die gleichzeitige Bearbeitung eines Projekts von mehreren Benutzern, dabei wird im Master-Repository Sperren auf Paketebene vergeben. Jeder neue Benutzer muß sich bestimmte Pakete aus dem Repository zuordnen lassen, auf die er dann Schreibzugriffe vornehmen kann. Alle anderen Informationen aus dem Repository können von diesem Benutzer nicht verändert werden. Hier kann es zu Unverträglichkeiten der einzelnen Benutzer-Repositorys kommen, die jedoch beim Einpflegen in das Master Repository vom System behoben werden. Dabei können eventuell Transaktionen rückgängig gemacht werden. Das Master Repository bleibt immer konsistent.

Codegenerierung für C++, Java und SQL

Der Codegenerator erzeugt C++ und Java-Code, sowie DDL-Code um die Anbindung an ein relationales Datenbanksystem zu unterstützen. Der Generator erzeugt das grundsätzliche Programmgerüst mit der Klassendeklaration, den Methodenrümpfen und den Attributen. Auf Wunsch generiert das Programm auch Standardmethoden zu jeder Klasse wie Konstruktor, Destruktor, Kopierkonstruktor, Zuweisungs- und Vergleichsoperator, sowie Zugriffsmethoden für alle Attribute. Leider ist es nicht möglich den Code einer Methode bereits innerhalb von OTW-2 zu erstellen. Der automatisch erstellte Code ist durch einige Kommentarzeilen ergänzt, in denen OTW-2 Zusatzinformationen speichert. Dies hält sich jedoch in Grenzen, sodaß der Code noch lesbar bleibt.

Das Reverse-Engineering funktioniert auch mit Codeteilen, die nicht vom Codegenerator erstellt wurden. Hierbei werden nur die Deklarationen ins Modell übernommen, das Programm merkt sich jedoch den zusätzlichen Code und fügt ihn auf Wunsch des Benutzers (Menüpunkt *Inkrementelles Generieren*) wieder in den neu generierten Code ein.

2.4 Paradigm Plus Enterprise 3.6

Hersteller	Platinum Technology Inc., Illinois, USA
Webseite	http://www.platinum.com/producs/appdev/pplus_ps.htm
Betriebssysteme	Windows 9x oder Windows NT, div. UNIX-Systeme

Positiv:

- ❑ Ausgereiftes Produkt mit großem Funktionsumfang, dadurch sowohl für kleinere als auch für größere Projekte effizient einsetzbar
- ❑ Bietet ausreichende Unterstützung von UML 1.1 mit Darstellung aller erforderlicher Diagrammtypen und den wichtigsten Notationselementen
- ❑ Lauffähig auch in heterogenen Umgebungen
- ❑ Starke Unterstützung von Teamarbeit und verteiltem Zugriff auf Repositorys durch verschiedenste Sicherungsmechanismen
- ❑ Repositorys als Object Store-Datenbank, dadurch hohe Effizienz und Möglichkeit der direkten Bearbeitung der Datenbasis
- ❑ *Inkrementelle* Codegenerierung, d.h. außerhalb des Werkzeugs geänderte Programmstücke werden erkannt und nicht überschrieben. Das Werkzeug kommt ohne spezielle Codemarkierungen aus
- ❑ Große Anzahl an Import- / Export-Schnittstellen
- ❑ Trotz des Umfangs zügiges Arbeiten möglich
- ❑ Übersichtliche Oberfläche, einfache Bedienung
- ❑ Gute Darstellung innerhalb des Grafikeditors
- ❑ Umfangreiche Dokumentation des Programms

Negativ:

- ❑ Keine automatische Konsistenzprüfung im Hintergrund, Ungereimtheiten beim Anlegen von Elementen (z.B. zwei gleichnamige Klassen), keine weiterführenden Erklärungen bei Konsistenzverletzungen
- ❑ Ausgabe von Dokumentationen in anderem Format als Text bzw. HTML nur über *add ons* realisierbar
- ❑ Effiziente Anbindung an relationale Datenbanken ebenfalls nur mittels Zusatzwerkzeug ERwin

Paradigm Plus bietet ausgereifte und umfangreiche Funktionen zum Entwerfen, Generieren und Verwalten von Applikationen verschiedenster Art.

Unterstützt mehrere Notationen

Im Bereich von objektorientierter Analyse und Entwurf unterstützt das Werkzeug relativ durchgängig UML 1.1 sowie weitere Notationen wie OMT, Booch, Fusion etc. und auch die OOCL-Methode (*Object Oriented Change and Learning*) zur Geschäftsprozeßmodellierung innerhalb eines Unternehmens. Im Bereich der UML können sämtliche enthaltenen Diagrammtypen bearbeitet werden, die Notationselemente werden weitgehend dem Standard entsprechend dargestellt. Freie Formen können ebenfalls eingezeichnet werden. Der Ausdruck von Diagrammen entspricht dem *WYSIWYG*-Prinzip, die üblichen Einstellungen, wie Hoch-/Querformat, Anpassung an die Seite, etc. werden über die Windows-Druckerschnittstelle abgewickelt.

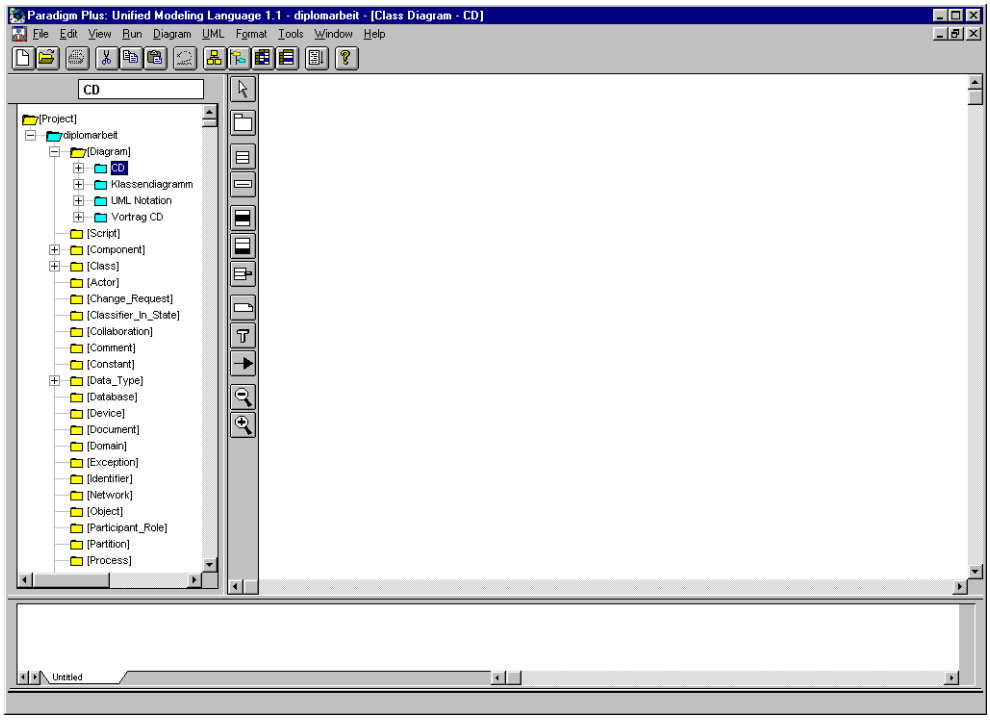

Abb. 2–7
Die Benutzeroberfläche von Paradigm Plus

Export als reiner Text bzw. HTML

Die in Paradigm Plus standardmäßig angebotenen Funktionen zur Reportausgabe umfassen zwei Formen, die Dokumentation des Modells in reinem Textformat bzw. der Web-Export in HTML, wobei auch Diagramme auf Wunsch als Grafikdateien eingebunden werden können und so eine umfassende Bereitstellung im Intra- und Internet ermöglichen. Die Generierung der Dokumente wird von benutzerdefinierten Scripts ausgeführt. Über das von Platinum zusätzlich angebotene Werkzeug ParaDoc können auf Basis von änderbaren Vorlagen noch umfangreichere Dokumentationen erstellt werden, die anschließend von gängigen Textverarbeitungen eingelesen werden können.

Schnittstellen zu anderen Produkten

Es bestehen Schnittstellen einerseits zu Produkten von Platinum selbst (*ParaDoc* u. a.), wie auch zu Werkzeugen anderer Hersteller, etwa Projekt-, Datenbank-, Konfigurations- und Versionsmanagementprogrammen, die mittels Import- und Exportfunktion, Kommando- oder auch API-Schnittstellen realisiert sind. Zur nahtlosen Anbindung an relationale Datenbanken steht das Zusatzprodukt ERwin zur Verfügung, welches die Daten des Objektmodells direkt auf ER-Diagramme abbilden kann.

Mehrbenutzerbetrieb auch in heterogenen Umgebungen

Die Daten eines Paradigm Plus-Projekts werden jeweils durch ein eigenes Repository verwaltet, welches als objektorientierte Datenbank im OODBMS Object Store implementiert ist. Im Sinne der Wiederverwendung besteht die Möglichkeit, auch Daten von anderen Repositorys zu referenzieren. Object Store wickelt auch die gesamte Mehrbenutzerkontrolle ab, dementsprechend kommen verschiedenste Sicherheits- und Wiederanlauf-Mechanismen zur Anwendung. Das Werkzeug ist sowohl in homogenen als auch in heterogenen Systemumgebungen lauffähig. Benutzer können im Mehrbenutzerbetrieb auf jedes verfügbare Repository zugreifen, Bearbeiter haben die Möglichkeit, unterschiedliche Sperren auf bestimmte Elemente zu legen und diese entweder nur zum Lesen bzw. auch zum Schreiben durch andere freizugeben. Im Bereich des Versionsmanagements ist Paradigm Plus direkt bzw. durch Anbindung an externe Produkte (wie PVCS, CCC/Harvest u.a.) in der Lage, verschiedene Versionen eines Projekts zu verwalten.

Codegenerierung in mehreren Zielsprachen

Die Codegenerierung erzeugt auf Wunsch die Coderahmen für verschiedene Zielsprachen. Unterstützt werden unter anderem (Visual) C++, Java, Smalltalk, Visual Basic, Delphi und unterschiedliche Datenbankmodelle. Dies wird ähnlich wie bei der Erzeugung von Reports über Scripts gesteuert, die den eigenen Anforderungen angepaßt werden können. Außerhalb des Werkzeugs geschriebene Programmteile werden erkannt und bei der Codegenerierung nicht überschrieben. Es ist im Rahmen des Reverse-Engineering möglich, Daten aus bereits bestehendem Code in das Repository zu übernehmen und automatisch die entsprechenden Diagramme zu erzeugen.

Über Scripts können verschiedene Analyse- und Entwurfsmetriken ausgegeben werden.

Gute Benutzeroberfläche

Die Benutzeroberfläche von Paradigm Plus entspricht aktuellen Windows-Standards und ist angenehm zu bedienen. Für die am häufigsten verwendeten Funktionen finden sich Symbole und Menüs, ohne daß der Bildschirm überfrachtet erscheint. Die Bedienung des Grafikeditors erfolgt im Großen und Ganzen intuitiv. Die Darstellung der Elemente eines Diagramms ist auch auf leistungsschwächeren Rechnern ausreichend schnell und auch die Möglichkeiten, ohne besonderen Aufwand ein ansprechendes Layout zu erstellen, funktionieren ohne Probleme. Hierzu zählen etwa Gitternetz-Ausrichtung der Symbole, automatische Layoutoptimierung oder die Anpassung von Farben, Strichstärken und Schriftgrößen. Die auf der Syntax von Visual Basic basierende Scriptsprache ist ohne große Probleme zu erlernen. Verschiedene Scripts können auch bestimmten Menüpunkten zugeordnet und damit bei Auswahl automatisch gestartet werden. Zum Bearbeiten der Elemente bzw. deren Eigenschaften stehen verschiedene Editoren zur Verfügung, so etwa der *Diagramm-*, *Formular-*, *Matrix-* und *Tabelleneditor*.

Die im Lieferumfang von Paradigm Plus enthaltene Dokumentation umfaßt Handbücher bzw. PDF-Dateien, die die gesamte Thematik abdecken (Start-, Benutzer- und Methodenhandbuch, sowie verschiedene Referenzen). In der Online-Hilfe können die entsprechenden Punkte der nichtgedruckten Bücher direkt angewählt werden.

2.5 Rational Rose 98 Enterprise SR-1

Hersteller	Rational Software Corp., Cupertino, USA
Webseite	http://www.rational.com/products/rose
Betriebssysteme	Windows 9x oder Windows NT sowie UNIX

Positiv:

- ❑ Recht gute UML-Unterstützung
- ❑ Umfangreiche Dokumentation (Hilfe, UML-Tutorium)
- ❑ Große Anzahl weiterführender Information (Internet, Literatur)
- ❑ Gute Benutzerschnittstelle, einfache individuelle Anpassung
- ❑ Vielversprechende Vorkehrungen für größere Projekte

Negativ:

- ❑ Automatische Layoutfunktion meist unbrauchbar
- ❑ Teilweise problematischer Mehrbenutzerbetrieb

- Überfrachtete Coderahmen
- Round-Trip Engineering relativ kompliziert
- Bestimmte UML-Konstrukte nicht standardkonform

Rational Rose ist eines der verbreitetsten und bekanntesten UML-Werkzeuge. Dies ist einerseits dadurch zu erklären, daß die "Väter" von UML, Grady Booch, Ivar Jacobson und James Rumbaugh bei Rational beschäftigt sind, andererseits dadurch, daß Rational Rose die vorgegebenen Konzepte (Diagrammtypen, Darstellung der Elemente, etc.) relativ standardkonform umsetzt. Neben UML werden von Rational Rose die Booch-Notation und OMT unterstützt; ein Wechseln zwischen verschiedenen Methoden ist einfach möglich.

Unterstützt auch die Booch-Methode sowie OMT

Das User-Interface von Rational Rose kann als sehr benutzerfreundlich bezeichnet werden. Dies fällt besonders auf, wenn man es mit anderen Werkzeugen vergleicht, welche eine ähnliche Komplexität wie Rose ausweisen. Die gängigen Windows-Standards, wie *Drag&Drop*, *OLE* u.a. werden weitgehend unterstützt. Dadurch wird der Lernaufwand zur Bedienung des Tools für Windows-Benutzer reduziert.

Gelungene Benutzerschnittstelle

Auch die bekannten GUI-*Styleguides* werden eingehalten. Rose präsentiert sich in einem Hauptfenster mit mehrfach geteilter Ansicht und frei konfigurierbaren Werkzeugleisten, zu allen wichtigen Operationen steht ein Kontextmenü mit allen wesentlichen Funktionsinhalten zur Verfügung, auch *Hot-Help* bzw. eine Statuszeile geben dem Benutzer weiterführende Informationen für sämtliche Bedienelemente.

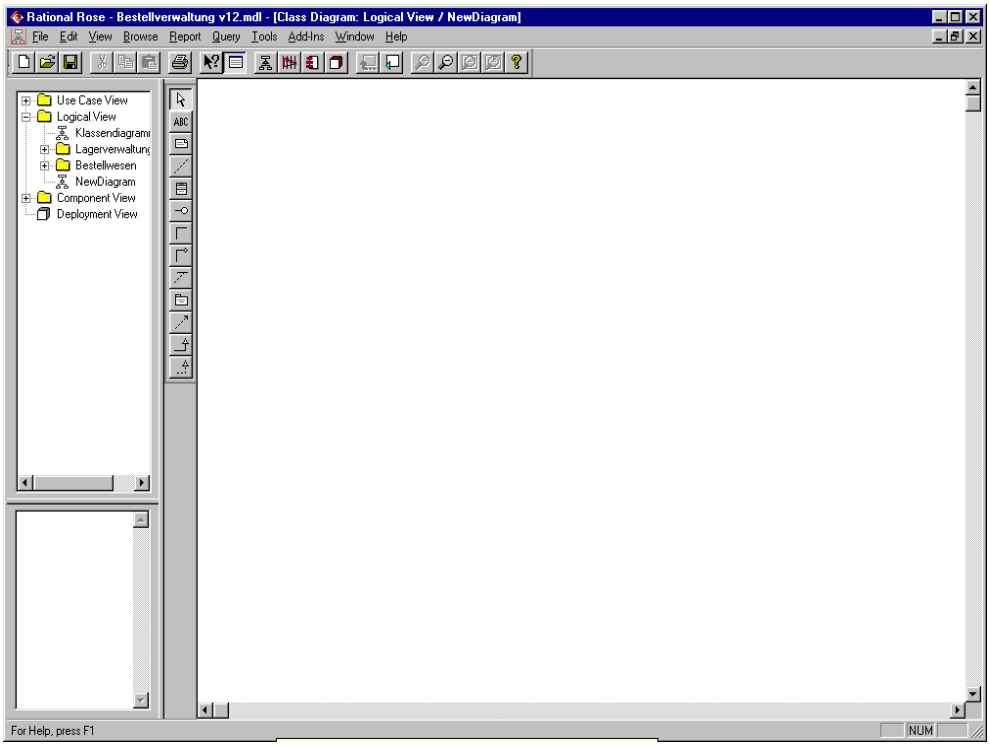

Abb. 2–8
Die Benutzeroberfläche von Rational Rose

Während Rational Rose in den Standard-Versionen auf *eine* Codegenerierungssprache beschränkt ist, bietet die *Enterprise Edition* Add-Ins, um die gebräuchlichsten Zielsprachen und -Umgebungen zu nützen (C++, Visual Basic, CORBA/IDL, Java), darüber hinaus ist durch die Exportfunktionen Oracle 8, ERwin und DDL auch eine Anbindung an die relationale Datenbankwelt gegeben.

Codegenerierung in vielen Zielsprachen

Über einen *Add-In-Manager* können sowohl grundlegende Scripts und Funktionen (z.B. von MS-Project oder auch von Werkzeugen zur Erstellung von Software-Metriken) als auch sprachspezifische (z.B. spezielle Codegenerierungsscripts) komfortabel in die Benutzerumgebung eingebunden werden. Es können sowohl Menüs, Hilfedateien, ausführbare Dateien, Scripts u.ä. verwendet werden.

Rational Rose gestattet es, über das Microsoft Repository Modellierungsinformationen mit anderen Werkzeugen und Programmen auszutauschen. Hierbei werden lediglich die im Repository enthaltenen Modellelemente, jedoch keine Diagramminformationen exportiert.

Mehrbenutzerbetrieb

Zur Unterstützung des Mehrbenutzerbetriebs wird ein Projekt in eigene separate Einheiten geteilt, sog. *controlled units*. Jeder einzelne Benutzer arbeitet somit in einer individuellen Sicht auf das gesamte Modell, das vor Änderungen geschützt bleibt. Auf Wunsch können diese den anderen Benutzern zur Verfügung gestellt werden. Dies erfolgt durch eine Anbindung an ein (leider nicht im Lieferumfang enthaltenes) Konfigurations-Management- und Versionskontrollwerkzeug, wie Rational ClearCase, ist jedoch durch den benötigten Intergrationsprozeß mit dem restlichen Modell auch mit gewissen Komplikationen verbunden.

Erzeugung von Reports über spezielle Scripts

Die Generierung von Reports wird wie bei anderen CASE-Tools über benutzerdefinierbare Scripts angestoßen. Die Scriptsprache ist Visual-Basic ähnlich. Im Lieferumfang sind verschiedene Reportscripts enthalten, die wichtigsten sind sicherlich der Export nach MS-Word bzw. der HTML. Auch auf der Webseite des Herstellers bzw. des Rational-Rose-Magazins *Rose Architect* werden weitere Scripts von Rose-Anwendern zur Verfügung gestellt. Hiermit können besonders die Reports, die standardmäßig nicht zu den umfangreichsten zählen, besser den individuellen Anforderungen angepaßt werden.

Wie bereits erwähnt, stellt die Enterprise-Edition eine Vielzahl von Zielsprachen zur Codegenerierung bzw. zum *round trip engineering* zur Verfügung. Über ein Kontextmenü kann für eine Komponente bzw. für eine einzelne Klasse ein Codegerüst in der für das jeweilige Element festgelegten Zielsprache erstellt werden. Hierbei werden wieder entsprechende Scripts verwendet, um einen gewünschten Detaillierungsgrad zu erreichen. Leider ist jedoch festzustellen, daß in diesem Zusammenhang ein für gewöhnliche Anwendungsbereiche zu überfrachteter Coderahmen erzeugt wird, der durch eine Vielzahl an Kommentaren und anderen Inhalten unleserlich wird und dadurch schwer weiterzubearbeiten ist.

Überfrachteter Code

Reverse-Engineering mächtig, aber kompliziert zu bedienen

Auch das *Reverse-Engineering*, d.h. das Rückführen von Code in ein Klassendiagramm gestaltet sich erheblich schwieriger als in anderen untersuchten Werkzeugen. Der sog. *C++-Analyzer* (siehe Abb. 2–9) ist zwar ohne Frage ein mächtiges Werkzeug mit umfangreichen Funktionen, jedoch fehlt die Möglichkeit, ohne großen Aufwand effizient das gewünschte Ergebnis zu erhalten. Außerdem ist durch die Tatsache, daß es sich hierbei um ein eigenes Programm handelt, der Eindruck einer direkten Integration des Kreislaufs zwischen Codegenerierung und Reverse-Engineering nicht gegeben.

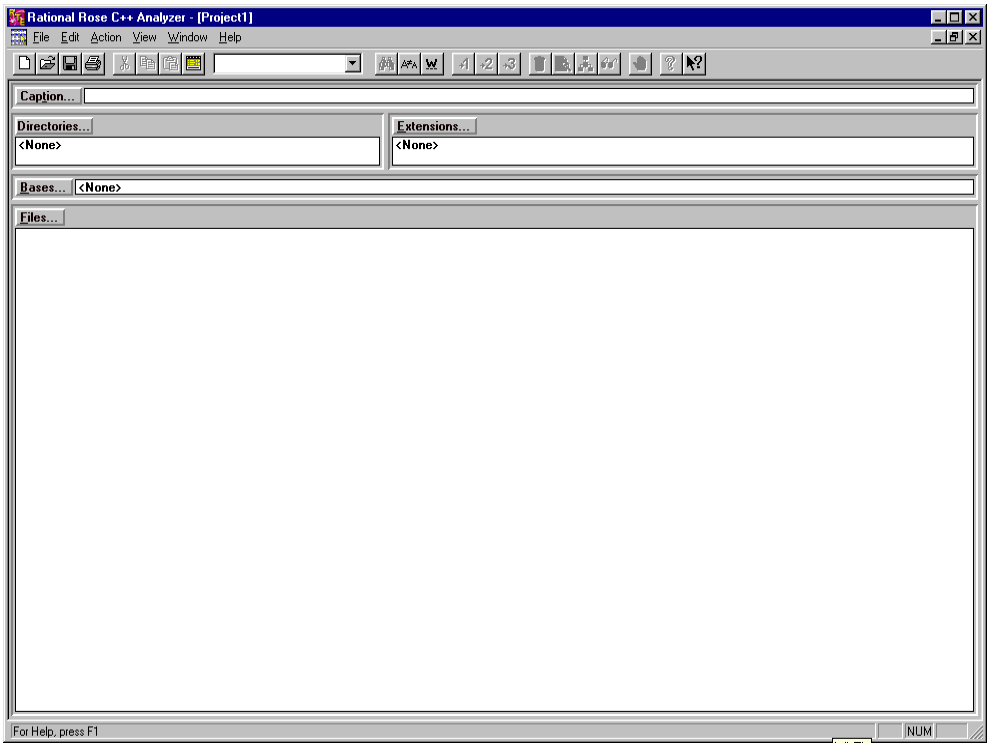

Abb. 2–9
Der Rational Rose C++ Analyzer

Rational Rose stellt Tastatur-Shortcuts für fast alle gängigen Aktionen zur Verfügung. Hierbei werden bei Standard-Windows-Kommandos auch die üblichen Shortcuts verwendet (Ausschneiden, Kopieren ...).

Die Druckfunktion für Diagramme, die das Programm bereitstellt ist ordentlich implementiert. Es kann in geringem Rahmen die Genauigkeit und einige wenige zusätzliche Details (Kopf- und Fußzeile) beeinflußt werden. Positiv ist die Tatsache, daß große Diagramme gekachelt auf mehrere Druckseiten verteilt werden können.

Umfangreiche Hilfe und weiterführende Informationen

Das Hilfesystem von Rational Rose basiert auf umfangreichen Windows-Hilfedateien, die gut gegliedert sind und viele Links zu verwandten Themen enthalten. Auch im Internet sind viele Informationen zu finden, die dem Benutzer bei Problemen helfen oder Tips geben können. Newsgroups und Online-Foren werden sowohl von der Herstellerfirma, als auch von Dritten zur Verfügung gestellt. Weiters wird mit Rational Rose ein sogenanntes *UML-Tutorial* ausgeliefert, das die grundlegenden Konzepte der UML erklärt.

2.6 SELECT Enterprise 6.0e SR-4

Hersteller	Princeton Softech Inc., Princeton, USA
Webseite	http://www.princetonsoftech.com/products/ps-select-select_enterprise.htm
Betriebssysteme	Windows 9x oder Windows NT

Positiv:

- ❑ Datenhaltung über OODBMS mit offener, skalierbarer Architektur und effizienter Mehrbenutzerkontrolle
- ❑ Automatisierte Synchronisation von Modell und Code
- ❑ Weitreichende Einstellungsmöglichkeiten der Codegenerierung direkt im Werkzeug definierbar
- ❑ Offengelegte OLE-Schnittstelle zum Datenaustausch mit anderen Anwendungen
- ❑ Funktionen zur Wiederverwendung von Komponenten in mehreren Projekten
- ❑ Ausführliche Konsistenzprüfung im Hintergrund
- ❑ Simulation des dynamischen Modells mittels *Object Animator*
- ❑ Umfangreiche Suchfunktionen
- ❑ Generierung detaillierter Reports

Negativ:

- ❑ Überfrachtete Benutzeroberfläche
- ❑ Beeinträchtigung der Usability durch große Anzahl von Dialogfenstern
- ❑ Bestimmte UML-Konstrukte nicht darstellbar
- ❑ Kein direkter HTML-Export möglich
- ❑ Werkzeug selbst nur unter Windows-Systemen lauffähig
- ❑ Modellierung der Systemdynamik (Sequenzdiagramm) zu kompliziert

SELECT Enterprise ist ein Werkzeug, das vor allem für große, verteilte Projekte ausgelegt ist. Eine der hervorstechenden Stärken liegt in der effizienten Datenhaltung der Modellierungsinformationen. Hierfür wird die objektorientierte Datenbank Softlab Enabler eingesetzt, die eine offene, vollständig skalierbare Architektur zur Verfügung stellt, mit leistungsfähigen Schnittstellen, über die auf die enthaltenen Daten von außen zugegriffen werden kann. Obwohl das Werkzeug selbst nur unter Windows 9x bzw. NT lauffähig ist, kann das Repository innerhalb eines Netzwerks auch von UNIX-Rechnern bereitgestellt werden. Komponenten anderer Projekte können in SELECT Enterprise direkt referenziert werden, sodaß eine Wiederverwendung effizient unterstützt wird.

Datenhaltung über OODBMS

Die Kontrolle des Mehrbenutzerbetriebs wird ebenfalls über Enabler abgewickelt. So können mehrere Arten von Sperren (sog. *Locks*) auf Modellebene eingerichtet werden, um einen Zugriff von außen zu verhindern. Es besteht auch die Möglichkeit, verschiedene Gruppen von Benutzern anzulegen, die mit unterschiedlichen Zugriffs- und Verwaltungsrechten ausgestattet sind.

Effizienter Mehrbenutzerbetrieb

Die graphische Benutzeroberfläche wurde von den Entwicklern sehr detailliert an die gewohnten Windows-Styleguides angepaßt. So steht etwa ein umfangreiches Browser-Fenster zur Verfügung, in dem verschiedene Modellsichten, sowie die ausführliche Online-Hilfe durchsucht werden können. Weitere Details umfassen z.B. die effektive Nutzung von Kontextmenüs sowie

GUI ansprechend implementiert

eine gut durchdachte Fenstertechnik. Selbst in Anbetracht der Komplexität des Werkzeugs ist somit ein relativ intuitives Arbeiten möglich.

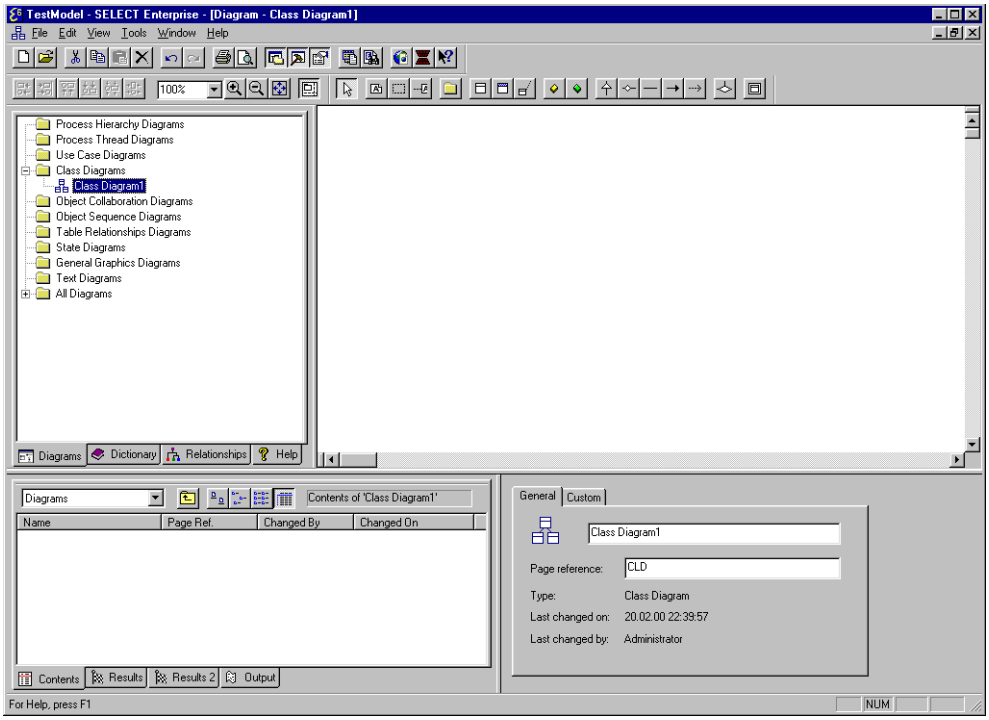

Abb. 2–10
Die Benutzeroberfläche von SELECT Enterprise

Usability etwas unbefriedigend

Leider wirkt der Arbeitsbereich jedoch relativ überladen und die eigentliche Diagrammansicht ist von zu geringer Größe. Auch die Eigenschaftenfenster sind, wahrscheinlich um eine bessere Strukturierung zu erreichen, in mehrere Karteikarten unterteilt - dadurch wird jedoch eine effiziente Eingabe der jeweiligen Informationen erschwert. So muß vom Benutzer meist ein Kompromiß in Bezug auf die Ergonomie bei der Bedienung des Werkzeugs geschlossen werden.

Die Umsetzung der UML-Konstrukte ist bis auf wenige Ausnahmen standardkonform, die automatisch im Hindergrund laufende Konsistenzprüfung verhindert im Vorhinein fehlerhafte Benutzereingaben. Der Zusammenhang zwischen den Modellen ist stets gewährleistet, so muß etwa ein Zustandsdiagramm immer einer bestimmten Klasse oder ein Sequenzdiagramm dem jeweiligen Anwendungsfall (*Use-Case*) zugeordnet werden.

Umfangreiche Codegenerierung

Aus dem statischen Systemmodell läßt sich Quellcode in verschiedenen Zielsprachen erzeugen (C++, Java, u.a.). Die Funktionalität des Codegenerators ist über entsprechende Menüeinträge umfangreich und bequem einstellbar, sämtliche relevanten Informationen werden korrekt übersetzt. Codestücke, die außerhalb des Werkzeugs hinzugefügt werden, bleiben bei einer erneuten Vorwärtsgenerierung erhalten und können auch in das Modell übernommen werden. Auch die Modellierung bzw. Erzeugung eines relationalen Datenbankschemas ist im Werkzeug selbst möglich. Über einen *Mapping*-Mechanismus können die Informationen des Objektmodells auf ein solches Schema direkt abgebildet werden, um anschließend SQL-Code zu erzeugen.

Automatisierte Synchronisation von Modell und Code

Eine besonders ansprechende Lösung wurde für das Reverse-Engineering bzw. die Erhaltung eines konsistenten Zustands zwischen Modell und dazugehörendem Quellcode implementiert: Ein sog. *Synchronizer* (siehe Abb. 2–11) stellt nach einem Vergleich der beiden Systembestandteile sämtliche Unterschiede graphisch dar und erlaubt ein automatisches Aktua-

lisieren der gewünschte Elemente. Auch hier werden die relevanten Konstrukte von SELECT Enterprise erkannt und korrekt (z.B. im jeweiligen Klassendiagramm) umgesetzt.

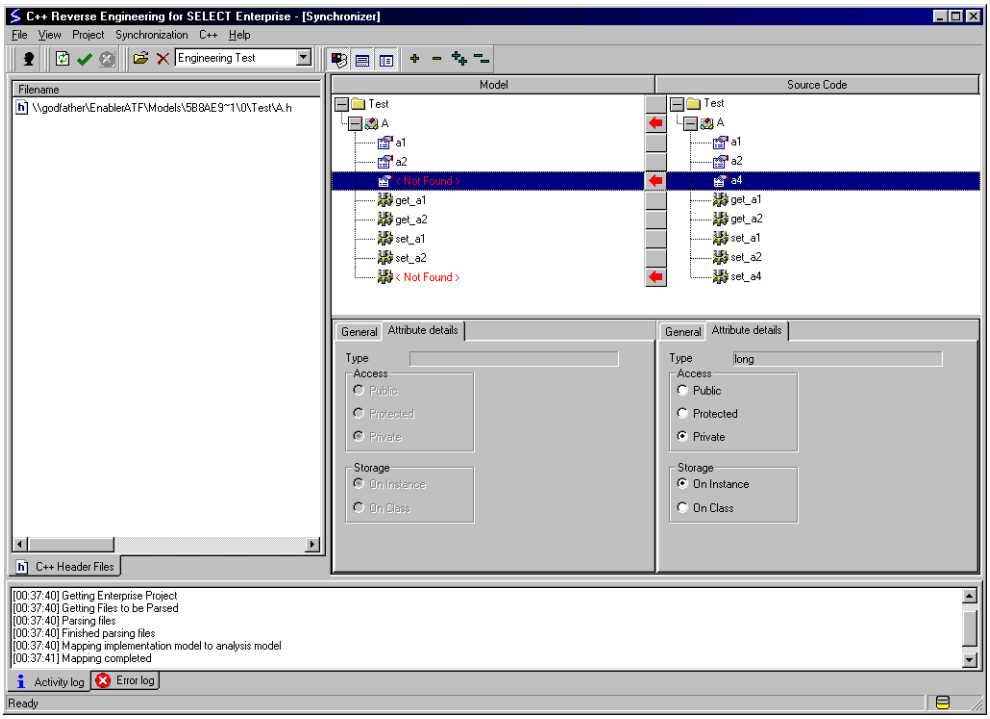

Abb. 2–11
Der SELECT Enterprise Synchronizer

Erstellung von Reports

Für die Ausgabe von Reports über das bearbeitete Projekt kommt einerseits der *Report-Writer* zum Einsatz, der rein textuelle Beschreibungen erzeugt, auf der anderen Seite bietet der *Document-Generator* über vordefinierte Schablonen die Möglichkeit, umfangreiche und gut aufbereitete Dokumentationen direkt nach MS-Word zu exportieren. Der Inhalt der entsprechenden Dokumente kann vom Benutzer einfach den Anforderungen angepaßt werden. Ein Export nach HTML ist leider nur über den Umweg der integrierten OLE-Schnittstelle realisierbar.

SELECT Enterprise kann sehr effizient in bestehende Entwicklungsumgebungen integriert werden, außerdem besteht die Möglichkeit, die Funktionalität des Werkzeugs durch andere Produkte der Herstellerfirma zusätzlich zu erweitern.

2.7 Software through Pictures/UML 7.1

Hersteller	Aonix Software, San Diego, USA
Webseite	http://www.aonix.com/content/products/stp/uml.html
Betriebssysteme	Windows 9x oder Windows NT sowie UNIX

Positiv:

- ❑ Gute Unterstützung der UML
- ❑ Offene Architektur mit umfangreichen Schnittstellen
- ❑ Leistungsfähige Datenhaltung
- ❑ Genaue und vollständige Umsetzung der Modellinformationen in Quellcode
- ❑ Effiziente Unterstützung des Mehrbenutzerbetriebs
- ❑ Lauffähig auch in verteilten, heterogenen Umgebungen
- ❑ Ausführliche Dokumentation
- ❑ Umfangreiche Scriptsprache

Negativ:

- ❑ Kein graphischer Browser, Suchfunktionen im Repository nur über Datenbankabfragen realisierbar
- ❑ Konsistenzprüfung muß eigens angestoßen werden
- ❑ Großteil der Werkzeuganpassungen nur über Veränderung der entsprechenden Scripts durchzuführen
- ❑ Ungereimtheiten beim Reverse-Engineering
- ❑ Hoher Ressourcenverbrauch, langsamer Bildschirmaufbau

Software through Pictures ist eine Familie von Softwareentwicklungswerkzeugen, die jeweils auf eine bestimmte Notation/Methode ausgerichtet sind. Es werden StP/OMT, StP/Booch, StP/IM zur Modellierung relationaler Datenbankschemata, sowie StP/SE für die strukturierte Analyse/Design-Methodik, sowie das hier getestete StP/UML angeboten.

Erste native NT-Version

Mit der StP/UML Version 7 ist dieses Werkzeug erstmals in einer nativen Windows-NT Version verfügbar, zuvor war das Werkzeug nur über eine Emulation auf anderen Systemen als UNIX lauffähig. Obwohl ein Großteil der Funktionalität an die einer herkömmlichen Windows-Anwendung angepaßt wurde, so sind dennoch einige Einschränkungen zu bemerken. Vor allem der Graphikeditor bietet einen nur ungenügend schnellen Bildschirmaufbau, auch die *Refresh*-Funktion (Neuzeichnen des Diagramminhalts) muß vom Benutzer meist mehrmals hintereinander ausgeführt werden, um nach bestimmten Veränderungen wirklich alle Elemente korrekt darzustellen.

Effiziente Datenhaltung

Zur Verwaltung des Repository kommt das relationale Datenbanksystem Sybase System 11 SQL Server zum Einsatz. Dieses bietet eine eindrucksvolle Funktionalität zur Verwaltung der Modelldaten, auch bei einer großen Anzahl von umfangreichen, verteilten Projekten. Jedoch ist der Ressourcenbedarf dementsprechend groß, sodaß bei einer Installation des Werkzeugs auch das Microsoft Jet Repository als Basis eingerichtet werden kann, das zwar eine verminderte Funktionalität (etwa keine Mehrbenutzerkontrolle) besitzt, jedoch keine derart großen Hardwareanforderungen an das System stellt. Durch den modulartigen Aufbau von StP ist der Modellierungsteil (in diesem Fall StP/UML) nur über bestimmte Schnittstellen mit der Datenbasis gekoppelt, wodurch eine Erweiterung oder Änderung des Systems einfach möglich ist (z.B. Wechsel von OMT nach UML etc.) und das Repository völlig unabhängig die entsprechenden Aufgaben zur Verwaltung der Modellinfor-

mationen übernehmen kann. Darunter fallen etwa Vermeidung von Redundanz, Backup, Recovery, Logging u.a.

Fehlen eines Repository-Browsers

Im Zusammenhang mit der Datenhaltung ist auch das Fehlen eines geeigneten graphischen Repository-Browsers zu erwähnen, wie ihn andere Entwicklungswerzeuge zur Verfügung stellen. Die einzige Möglichkeit, die Datenbasis nach bestimmten Modellelementen zu durchsuchen, besteht in der Formulierung entsprechender Datenbankabfragen (ähnlich SQL). Dies erfordert vom Anwender einen zusätzlichen Einarbeitungsaufwand und behindert das flüssige Arbeiten im Projekt.

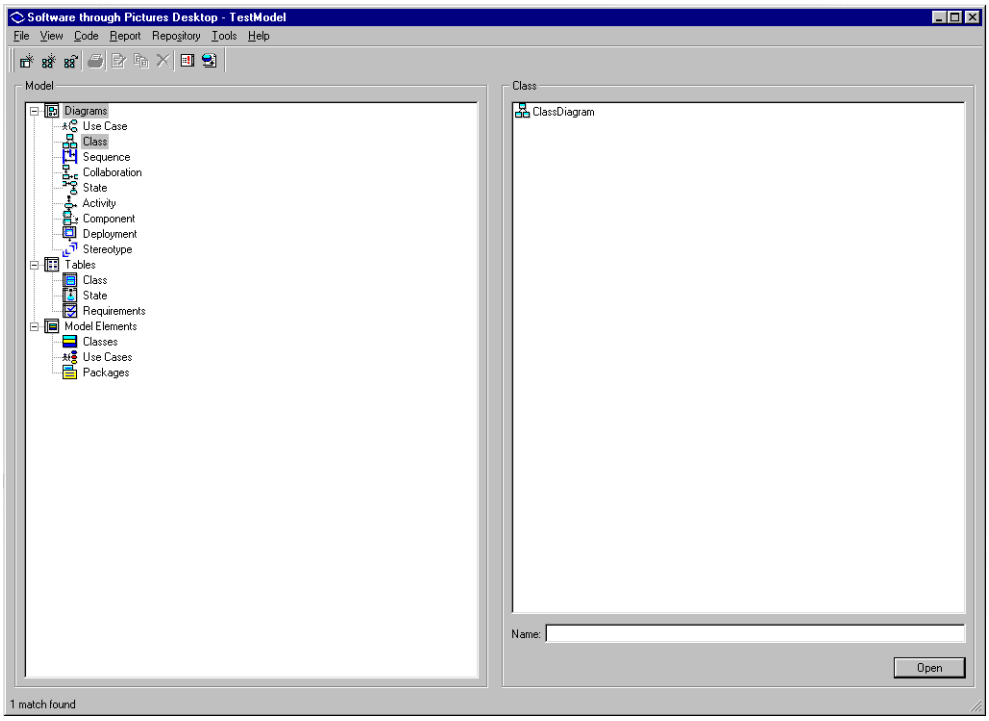

Abb. 2–12
Das Hauptfenster von StP/UML

Mehrbenutzerbetrieb

StP/UML ist auf unterschiedlichsten Rechnersystemen lauffähig (Windows NT, versch. UNIX-Systeme) und daher sehr stark für den Mehrbenutzerbetrieb in verteilten Umgebungen ausgelegt. Die Verwaltung der einzelnen Benutzer und deren Rechte (Konfliktlösungsmechanismus, *Locking* etc.) wird hierbei vollständig an das Datenbanksystem delegiert. Die entsprechenden Einstellungen können jedoch direkt im Werkzeug vorgenommen werden (Administration, Paßwortvergabe, Anzeigen und Ändern der Zugriffssperren etc.).

Umfangreiche Scriptsprache

StP/UML verfügt über eine leistungsfähige Scriptsprache, durch die ein Großteil der Werkzeugeinstellungen spezifiziert werden kann. Dies reicht von Art und Umfang der Codegenerierung, Konsistenzprüfung, Erzeugung von Reports bis zu Einstellungen der Benutzeroberfläche (Tastaturkürzel, Symbole etc.). Der Nachteil liegt hierbei einerseits darin, daß immer wiederkehrende Einstellungen jeweils nur umständlich über eine Änderung bestimmter Scripts durchführbar sind, andererseits leidet die Performanz des Werkzeugs zusätzlich, da etwa beim Starten eines Diagrammeditors erst die entsprechenden Regeldateien eingelesen werden müssen.

Codegenerierung für C++, Java, IDL, Ada und Forté TOOL

Wie bereits erwähnt wird die Codegenerierung über verschiedene Scripts spezifiziert. StP/UML kann aus dem statischen Modell Quellcode in verschiedenen Zielsprachen erzeugen, u.a. C++, Java, IDL, Ada und Forté TOOL. Dieser repräsentiert vollständig die relevanten Informationen der Modellstruktur und ist überdies nicht mit Kommentaren überfrachtet, was die Les-

barkeit und Erweiterbarkeit unterstützt. Das Werkzeug benutzt spezielle Markierungen im Quellcode, um bei einer erneuten Generierung Teile, die vom Benutzer außerhalb von StP/UML hinzugefügt wurden nicht zu überschreiben. Es besteht jedoch auch die Möglichkeit, innerhalb des Werkzeugs einzelnen Elementen (z.B. Operationen) bereits Codestücke hinzuzufügen, diese werden beim Generierungsvorgang ebenfalls umgesetzt.

Das Wiedereinlesen von Code (Reverse-Engineering) gestaltet sich ebenso vollständig, es sind jedoch einige kleine Ungereimtheiten zu erkennen (siehe Beispiel in Kapitel 5, *Reverse-Engineering*, S. 135).

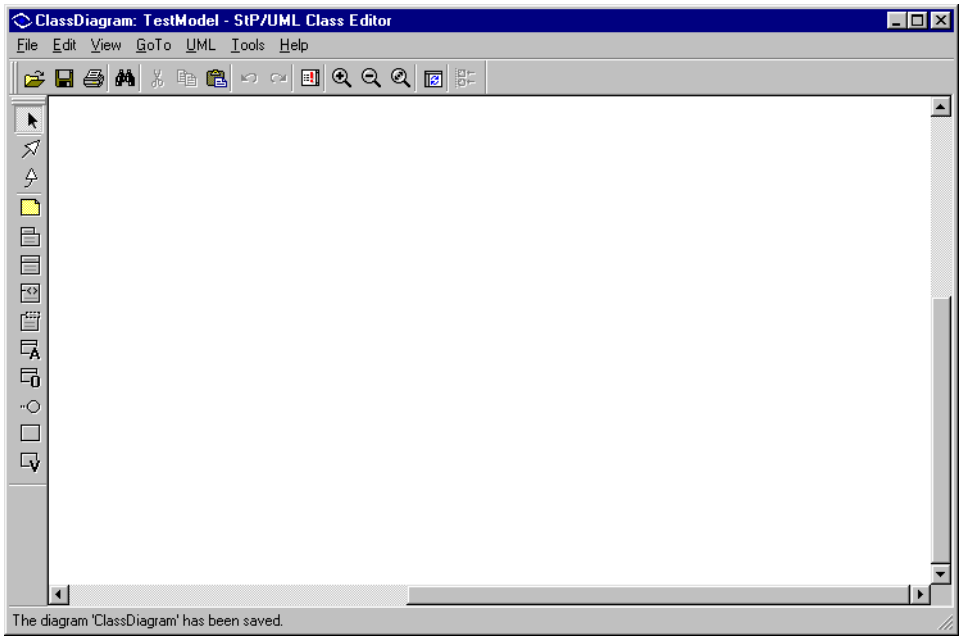
ClassDiagram: TestModel - StP/UML Class Editor
File Edit View GoTo UML Tools Help
The diagram 'ClassDiagram' has been saved.

Abb. 2–13
Der Diagrammeditor von StP/UML

Gute Dokumentation des Werkzeugs

Die mit StP/UML gelieferte Dokumentation ist sehr umfangreich und vollständig und deckt alle wichtigen Bereiche, von der Anleitung zur UML-Modellierung bis zu technischen Details der Administration und Verwendung des Werkzeugs ab. Da die Dokumente, die im PDF-Format vorliegen jedoch eine gewohnte Online-Hilfe ersetzen, ist eine effiziente Suche nach den gewünschten Themen oder Stichworten oft nicht möglich.

Detaillierte Reports über das aktuelle Projekt können sowohl direkt aus dem Werkzeug heraus erstellt werden (ein Export nach RTF, HTML, FrameMaker sowie ASCII wird unterstützt), darüberhinaus stehen Schnittstellen zu externen Dokumentationswerkzeugen zur Verfügung.

2.8 WithClass 99 Enterprise

Hersteller	Microgold Software Inc., New Jersey, USA
Webseite	http://www.microgold.com/version2/stage/product.html
Betriebssysteme	Windows 9x oder Windows NT

Positiv:

- ❑ Geringe Anforderungen an die Hardware
- ❑ Intuitive und unkomplizierte Bedienung
- ❑ Codegenerierung und Reverse-Engineering funktionieren problemlos und sind leicht konfigurierbar
- ❑ Möglichkeit der Eingabe von eigenen Programmstücken direkt im Werkzeug, auch Weiterbearbeitung von bestehendem Code
- ❑ VBA-Einbindung, dadurch Möglichkeit zur Erstellung von eigenen GUIs bzw. benutzerspezifischen Makros
- ❑ Große Breite an Zielsprachen
- ❑ Darstellung der Elemente bis auf wenige Ausnahmen UML-konform
- ❑ Sehr gute und umfangreiche Online-Hilfe
- ❑ Einfache Scriptsprache, dadurch sind sowohl Codegenerierung als auch die Erzeugung von Dokumenten leicht anzupassen

Negativ:

- ❑ Teilweise chaotische Darstellung der Elemente im Grafikeditor, keine automatischen Layoutfunktionen
- ❑ Zusammenhang zwischen den Modellen ist marginal
- ❑ Sehr schlechte Performanz bei der Codegenerierung
- ❑ Schwache Export-Unterstützung
- ❑ Mehrbenutzerbetrieb und Sicherheitsfunktionen für große verteilte Projekte unzureichend
- ❑ Datenhaltung über MS-Access Datenbank bei großem Datenvolumen wenig effizient
- ❑ Bescheidener Funktionsumfang, daher nur für kleinere Projekte sinnvoll einsetzbar

Einfache Bedienung vs. eingeschränkter Funktionsumfang

WithClass ist zu den Werkzeugen mit eingeschränkten Funktionsumfang zu zählen, die sich daher besonders für kleinere Projekte eignen. Seine Stärken liegen in der einfachen und intuitiven Bedienung und seinen geringen Anforderungen an Hardware und Systemumgebung. Ohne großen Aufwand an Einarbeitung in die Funktionen können so die Phasen von Analyse und Entwurf eines Modells durchschritten werden. Eine Besonderheit stellt die Tatsache dar, daß WithClass das erste Werkzeug am Markt ist, welches über eine VBA-Anbindung verfügt. Dadurch ist es möglich, nicht nur wiederkehrende Arbeitsschritte über Makros zu automatisieren, sondern auch eigene GUIs, ActiveX-Controls u.ä. zu generieren und damit speziellen Benutzerwünschen anzupassen.

Unterstützt mehrere Notationen

Der Diagrammeditor selbst stellt die vorgesehenen Modellelemente größtenteils UML-getreu dar, weiters ist die Darstellung in den Notationen OMT, Booch, Martin-Odell u. a. möglich. Auch freie Formen wie Rechteck oder Ellipse können in das Diagramm eingezeichnet werden. Bei der Bearbeitung eines Modells ist die Abbildung des logischen Zusammenhangs zwischen den einzelnen Modellen nicht immer auch im Werkzeug eindeutig definierbar. Im Gegensatz zu früheren Versionen ist es jetzt auch möglich, über einen Dia-

grammbrowser einfach zwischen den einzelnen Diagrammarten zu navigieren. Leider ist der Grafikeditor auch die offensichtlich größte Schwäche von WithClass. Es ist dem Benutzer kaum möglich, das Layout ansprechend zu gestalten, ohne daß nicht etwa Beziehungen zwischen Klassen sich gegenseitig überschneiden oder durch andere Elemente sichtbar hindurchführen.

Diagrammeditor sehr unbefriedigend

Die Ausgabe von Reports wird von verschiedenen Scripts gesteuert, auch über die bereits erwähnten VBA-Makros können diese an die Erfordernisse angepaßt werden. Die erzeugten Dokumente liegen standardmäßig als ASCII-File vor, es können jedoch auch HTML- und MS-Word Dateien (MS-Word über VBA-Makros) erzeugt werden, die eine Aufstellung der im statischen Modell enthaltenen Elemente beinhalten.

Die Daten eines Projekts werden in zwei verschiedenen Dateien gespeichert und können nach Bedarf über ein eigenes Repository verwaltet werden, das als MS-Access-Datenbank zur Verfügung steht. Dieses kann auch Modellierungsinformationen von mehreren Projekten beinhalten, wodurch Synchronisation bzw. Wiederverwendung von Komponenten ermöglicht wird. Das Verwenden eines Versionsmanagements wird durch die Anbindung an den PVCS-Version-Manager gewährleistet. Auch ein Arbeiten in Teams, wobei jeder Benutzer auf dieselbe Datenbasis zurückgreift, ist möglich, jedoch nicht in heterogenen Umgebungen, da WithClass nur auf Windows-Systemen lauffähig ist. Sicherheitsrestriktionen innerhalb des Werkzeugs, wie etwa die Vergabe von Paßwörtern, sind nicht vorhanden.

Repository als MS-Access Datenbank

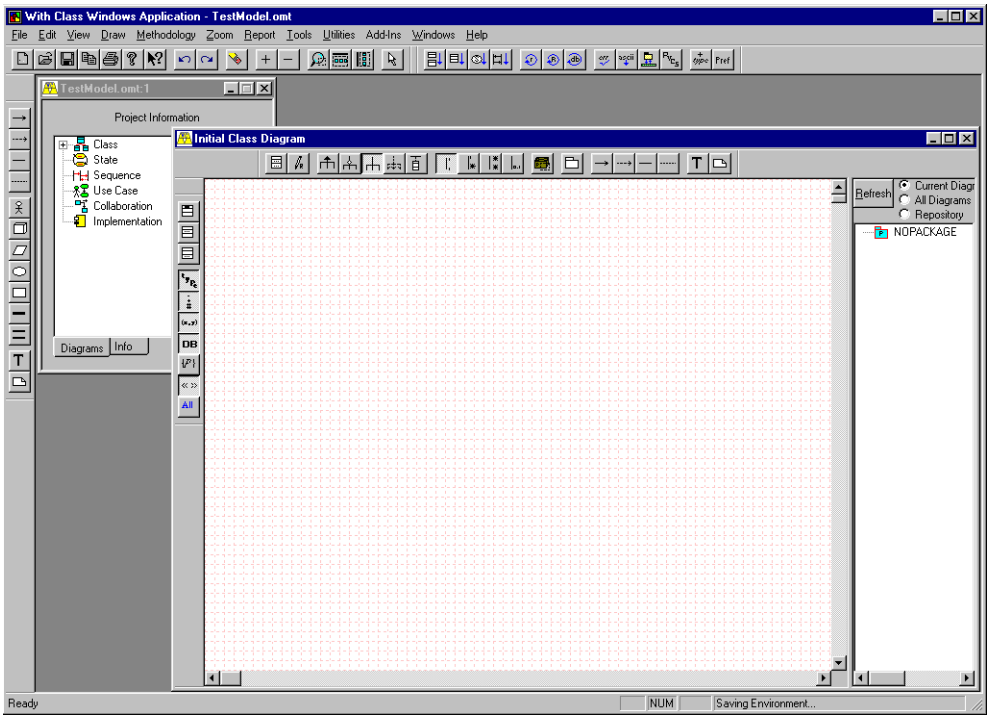

Abb. 2–14
Die Benutzeroberfläche von WithClass

Eine Konsistenzprüfung des Modells kann auf Wunsch durchgeführt werden, bietet dem Benutzer jedoch kaum weiterführende Informationen.

Die Codegenerierung gestaltet sich unkompliziert und flexibel. Über benutzerdefinierbare Scripts werden aus den Informationen des statischen Modells die Coderahmen für verschiedenste Zielsprachen erzeugt, unter anderem (Visual) C++, Visual Basic, Delphi, Java, Cobol, Smalltalk und Eiffel. Zur Erstellung von relationalen Datenbanken ist die Generierung von SQL-Code möglich. Es werden alle Elemente des Klassendiagramms in den Code übersetzt. Bereits innerhalb des Werkzeugs können über einen Editor über

Flexible, aber langsame Codegenerierung

den Rahmen hinausgehende Programmstücke angelegt werden, die dann in die entsprechende Quelldatei exportiert werden können. Wird Code außerhalb des Werkzeugs geändert, so kann dieser über eine Import-Funktion auch wieder eingelesen und im Werkzeug weiterbearbeitet werden. Leider ist die Geschwindigkeit der Codegenerierung im Vergleich zur Komplexität des Werkzeugs höchst unbefriedigend.

Reverse-Engineering

Reverse-Engineering, d.h. das Erstellen von Klassendiagrammen aus bereits bestehendem Programmcode wird gut unterstützt, ohne daß WithClass auf besondere Kommentare und Markierungen angewiesen ist. Die im Quellcode enthaltenen Informationen werden korrekt und vollständig im statischen Modell umgesetzt, die Performanz ist in diesem Bereich mehr als ausreichend.

Ansprechende Benutzeroberfläche

Die Benutzeroberfläche entspricht den allgemeinen Windows-Styleguides, wirkt kaum überladen, bietet jedoch leider kaum eine Möglichkeit der Anpassung (Menüs, Symbolleisten). Auch die erweiterten Funktionen von Windows (Kontextmenü über rechte Maustaste, OLE, etc.) werden effizient genutzt.

Die Dokumentation zu WithClass besteht aus zwei Teilen, einer recht umfangreichen Online-Hilfe und einem Handbuch im PDF-Format, welches aber nur die allgemeinsten Funktionen beschreibt. Weiters werden vom Hersteller Microgold auf dessen Webseite verschiedene *online*-Tutorien angeboten, die jedoch in ihrem Informationsgehalt stark variieren.

2.9 Zusammenfassung

Aufgrund der unterschiedlichen Komplexität und Funktionalität der in dieser Arbeit bewerteten Produkte ist es schwierig, eine eindeutige Empfehlung für ein bestimmtes Werkzeug zu geben. Vielmehr ist in der Praxis meist primär die Art und der Umfang aktueller bzw. zukünftiger Softwareentwicklungsprojekte ausschlaggebend für die Wahl eines Tools.

In Tab. 2–1 werden die Bewertungen der wichtigsten Aspekte, die in diesem Zusammenhang von Bedeutung sind, zusammenfassend gegenübergestellt.

Verwendete Symbole:

○○○	Aspekt wird unzureichend unterstützt
●○○	Aspekt wird schwach unterstützt
●●○	Aspekt wird gut unterstützt
●●●	Aspekt wird besonders gut unterstützt

Aspekt:	Cool:Jex	objectiif	OTW-2	Paradigm	Rose	SELECT	StP	WithClass
Umfang der Notationen	●●○	●○○	●○○	●●●	●●●	●○○	●○○	●●●
Datenhaltung / Repository	●●●	●●○	●●●	●●○	●○○	●●●	●●●	●○○
Mehrbenutzer-betrieb	●●●	●●○	●●○	●●●	●●○	●●●	●●●	○○○
Schnittstellen, Import/Export	●●○	●●○	●●○	●●●	●●●	●●○	●●○	●○○
Round-Trip Engineering	●●○	●●○	●●○	●●○	●●○	●●●	●●○	●●○
Benutzeroberfläche	●○○	●●○	●○○	●●○	●●●	●○○	●○○	●●○
Bedienungs-freundlichkeit	●○○	●●○	●●○	●●○	●●○	●●○	●○○	●●○

Tab. 2–1 *Gegenüberstellende Bewertung anhand ausgewählter Aspekte*

Außerdem haben die Autoren versucht, in Abb. 2–15 - abhängig von der Komplexität eines Softwareentwicklungsprojekts - einen Rahmen für den Einsatz der einzelnen Werkzeuge vorzuschlagen. Hierbei wird im Besonderen auf den Aufwand eingegangen, den die Programme bei einem Ersteinsatz verursachen. Es ist plausibel, daß auch die komplexeren Werkzeuge für kleine Projekte geeignet sind, wenn diese bereits gut in den Entwicklungsprozeß integriert sind.

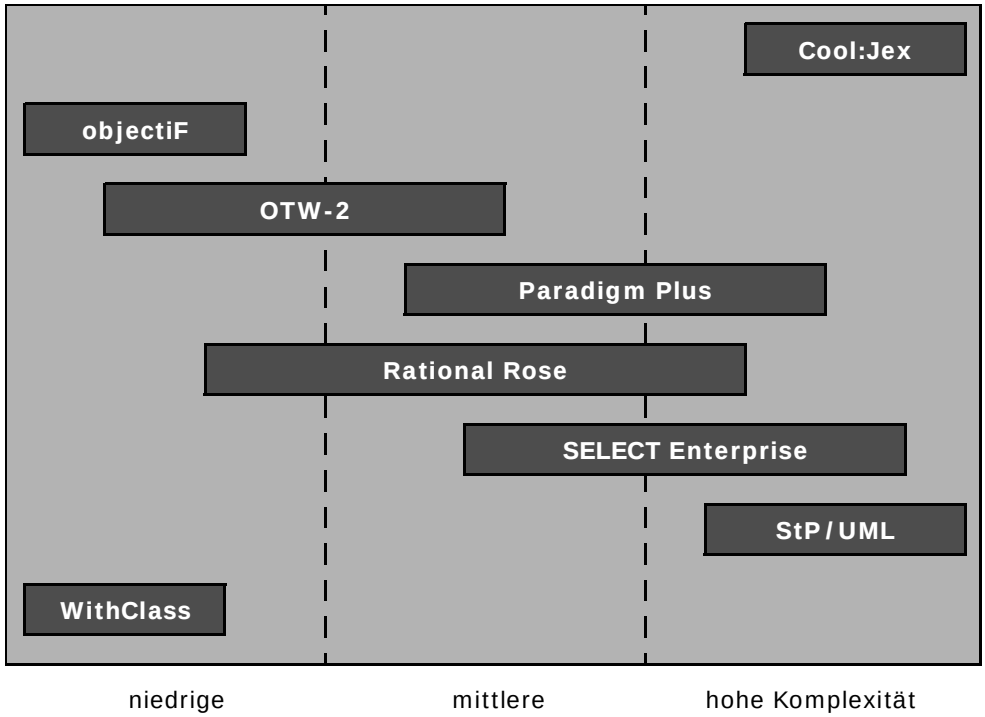

Abb. 2–15
Vorschlag für den Einsatz der Werkzeuge in Abhängigkeit von der Projektkomplexität

Aufgrund ihrer eingeschränkten Funktionalität sind die Programme objectiF und WithClass nur für eher einfache Projekte geeignet. WithClass etwa bietet in diesem Zusammenhang eine nur unzureichende Unterstützung des Mehrbenutzerbetriebs. Am Beispiel von objectiF kann gesehen werden, daß sich ein geringerer Funktionsumfang aber auch positiv auf die Benutzerfreundlichkeit und Einarbeitungszeit auswirken kann.

OTW-2 stellt eine Art Sonderfall dar. Die Komplexität liegt deutlich über der der beiden zuvor genannten Werkzeuge, jedoch entfernt sich das Programm zum Teil erheblich vom UML-Standard.

SELECT Enterprise und Paradigm Plus bieten einerseits umfangreiche Funktionalität für die Bearbeitung von großen Projekten in verteilten Umgebungen zusammen mit einer effizienten Mehrbenutzerkontrolle. Andererseits ermöglichen die beiden Programmen eine relativ einfache und intuitive Bedienung.

Das Werkzeug Rational Rose deckt den weitesten Bereich ab. Die Programmsteuerung ist bezogen auf die Komplexität sehr gut gelungen. Die etwas ineffiziente Datenhaltung (kein Datenbanksystem als Repository) schränkt jedoch den Einsatzbereich etwas ein.

Die beiden Programme Cool:Jex und StP/UML bieten zwar die größte Funktionalität, aufgrund der umständlichen und wenig intuitiven Bedienung erfordern diese Werkzeuge jedoch einen relativ hohen Aufwand für die Einarbeitung.

3 UML-Konformität in der Darstellung

Bezogen auf die UML-Notation in der Version 1.1 wird hier die Darstellung ausgewählter Konstrukte bewertet. Die einzelnen Modellsapekte sind den unterschiedlichen Diagrammarten zugeordnet. Umfaßt die Funktionalität eines Werkzeugs eine bestimmte Diagrammart, so ist dies mit dem Kürzel "J" gekennzeichnet, im Gegensatz zu "N", sollte das jeweilige Diagramm nicht dargestellt werden können.

Verwendete Symbole:

+	Darstellung sehr nahe bis ident dem UML 1.1-Standard
-	Darstellung entspricht kaum oder nicht der UML
n	Konstrukt kann nicht dargestellt werden

Modellaspekt:	Cool:Jex	objectiif	OTW-2	Paradigm	Rose	SELECT	StP	WithClass
Klassendiagramm	J	J	J	J	J	J	J	J
Anmerkung (Note)	+	+	+	+	+	+	+	+
Stereotype für Klassen	+	+	+	+	+	+	-	+
Stereotype für Beziehungen	n	n	+	+	+	n	-	+
Ikonenrepräsentation von Stereotypes	n	n	+	n	+	n	n	n
Package	+	n	+	+	+	+	+	+
Package-Inhalt	-	n	+	+	+	+	+	-
Klasse	+	+	+	+	+	+	+	+
Objekt	+	n	+	+	+	n	+	n
Abstrakte Klasse	+	-	+	+	+	+	+	+
Sichtbarkeit von Komponenten	+	+	+	+	+	+	+	+
Attribut	+	+	+	+	+	+	+	+
Methode	+	+	+	+	+	+	+	+

Tab. 3–1
UML 1.1 Konformität in der Darstellung

Modellaspekt:	Cool:Jex	objectiif	OTW-2	Paradigm	Rose	SELECT	StP	WithClass
Abstrakte Methode	+	n	+	-	n	-	-	-
Interface	+	+	+	+	+	n	+	-
Realizes-Interface-Beziehung	+	n	n	+	+	n	+	-
Schablone	+	+	+	-	+	-	+	-
Schabloneninstanzierung	+	n	+	+	+	n	+	-
Binäre Assoziation	+	+	-	+	+	+	+	+
Bezeichnung von Relationen	+	n	n	+	+	+	+	+
Rollenbezeichnung	+	+	+	+	+	+	+	+
Multiplizität	+	+	+	+	+	+	+	+
Qualifikation	+	n	n	-	+	+	+	-
Navigierbarkeit	+	+	+	n	+	+	+	-
Aggregation	+	+	+	+	+	+	+	+
Starke Aggregation (Komposition)	+	n	+	+	n	+	+	+
Aggregation: Geschachtelte Darstellung	n	n	n	n	n	n	n	n
Assoziationsklassen	+	n	n	+	+	+	+	+
N-äre Assoziation	+	n	n	n	n	-	-	n
Links	+	n	n	+	+	n	+	-
Generalisierung	+	+	+	+	+	+	+	+
Shared Target Style	+	n	n	+	+	+	n	+
Einschränkungen / Constraints	+	n	n	n	+	-	n	n
Dependency	+	n	+	+	+	+	+	+
Abgeleitete Elemente	+	n	n	n	+	n	-	n
Use-Case-Diagramm	J	J	J	J	J	J	J	J
Use-Case	+	+	+	+	+	+	+	+
Aktor	+	+	+	-	+	+	+	-
Use-Case-Beziehungen	+	+	+	+	+	+	+	+

Modellaspekt:	Cool:Jex	objectiif	OTW-2	Paradigm	Rose	SELECT	StP	WithClass
Sequenzdiagramm	J	J	J	J	J	J	J	J
Message	+	+	+	+	+	+	+	+
Aktivierungsbalken	+	+	+	n	+	+	+	+
Konstruktion/Destruktion von Objekten	+	+	+	n	-	n	-	n
rekursive Nachrichten	+	+	+	-	-	n	+	-
Kollaborationsdiagramm	J	N	N	J	J	J	J	J
Kollaboration als Pattern	+	n	n	+	n	n	+	-
Kontrollfluß	+	n	n	+	+	+	+	-
Zustandsdiagramm	J	J	N	J	J	J	J	J
Zustand	+	+	n	+	+	+	+	-
Anfangszustand	+	n	n	+	+	+	+	+
Endzustand	+	n	n	+	+	+	+	+
Aggregierter Zustand	+	n	n	+	+	+	+	-
History-Zustand	+	n	n	+	+	n	+	n
Tiefer History-Zustand	+	n	n	+	+	n	+	n
Synchronisationsbalken	+	n	n	n	n	+	n	+
Ereignis	+	n	n	+	+	+	+	-
Aktivitätsdiagramm	J	N	N	J	N	N	J	N
Entscheidung	+	n	n	+	n	n	+	n
Swimlane	+	n	n	+	n	n	n	n
Action-Object-Flow-Beziehung	+	n	n	+	n	n	+	n
Kontroll-Icons	+	n	n	n	n	n	n	n
Implementationsdiagramm	J	N	J	J	J	N	J	J
Deployment-Diagramm	J	N	J	J	J	N	J	N

4 Kriterienkatalog

Der für die folgende Auswertung herangezogene Kriterienkatalog basiert auf dem Inhalt eines Proseminars des *Instituts für Informatik und Wirtschaftsinformatik* der Universität Wien aus dem Jahr 1998, im Zuge dessen bereits eine Auswahl von UML-Werkzeugen evaluiert wurde. Der Katalog gliedert sich in drei wesentliche Abschnitte:

- Zuerst wird die Umsetzung der wichtigsten Konzepte der UML untersucht.
- Der zweite Abschnitt behandelt wesentliche technische Funktionen, wie Datenhaltung, Code-Generierung, Mehrbenutzerbetrieb, und anderes.
- Abschließend werden Punkte erörtert, die für die Bedienung der Programme von Bedeutung sind.

4.1 Methodenbasierte Kriterien

a) Modellierung der Objektstruktur - Objektstatik

1. Alle Basiselemente zur Beschreibung eines statischen Klassenmodells werden automatisch im zugrundeliegenden Data-Dictionary gespeichert.
2. Einmal erfaßte Elemente einer Klasse können jederzeit nachträglich verändert werden, sofern die Änderungen mit den bisher erfaßten Inhalten verträglich sind (z.B. Namenskonflikte).
3. Die Reihenfolge der Komponenten einer Klasse im Klassendiagramm kann der Benutzer bestimmen.
4. Für jedes Attribut kann ein Datentyp festgelegt werden.
5. Für eine Methode kann ein Rückgabetyp angegeben werden, falls eine Methode einen Wert bzw. ein Objekt zurückliefert.
6. Für eine Methode können Parameter angegeben werden, die beim Methodenaufruf übergeben werden. Dabei kann auch spezifiziert werden, ob es sich um Eingangs-, Ausgangs- oder Durchgangsparameter handelt.
7. Für eine Methode können Vor- und Nachbedingungen angegeben werden, die erfüllt sein müssen (*assertions*).

b) Modellierung der Systemstruktur - Systemstatik

1. Auch rekursive Aggregationsbeziehungen können erfaßt werden.
2. Einmal erfaßte Beziehungen können jederzeit nachträglich verändert werden, sofern die Änderungen mit den bisher erfaßten Inhalten verträglich sind.
3. Unabhängig von der Zuordnung von Klassen zu Packages lassen sich Klassen nach bestimmten Gesichtspunkten in Klassenkategorien einteilen.
4. Es werden OCL-Annotationen unterstützt.

c) Dynamisches Objektverhalten - Objektdynamik

1. Für jedes Objekt bzw. jede Klasse (Abstraktion) kann ein Zustandsdiagramm erstellt werden. Dieses ist umkehrbar eindeutig mit der jeweiligen Klasse verbunden.
2. Es können Bedingungen modelliert werden, die einen Zustandsübergang bewirken. Dabei ist ein Bezug auf Attribute des Objekts möglich.
3. Das Werkzeug kann Aktionen darstellen, die einem Zustandsübergang zugeordnet sind. Dabei ist ein Bezug auf Methoden des Objekts möglich.
4. Das Tool ermöglicht die Verfeinerung von Aktivitäten und Ereignissen im Sinne einer genaueren Beschreibung.
5. Die Darstellung von geschachtelten Zustandsdiagrammen, die eine Generalisierung von Zuständen modellieren, ist möglich. Dabei muß ein Objekt in irgend einem (exklusives oder) Zustand des untergeordneten Zustandsdiagramms (*Unterstatus*) sein.
6. Die Modellierung einer Generalisierung von Ereignissen ist möglich. Diese ist unabhängig von der modellierten Klassenhierarchie.

d) Dynamisches Systemverhalten - Systemdynamik

1. Die Darstellung einer Aggregation von Zuständen von nebenläufigen Objekten (*aggregation concurrency*) ist möglich. Dabei ist jedes Objekt dieser Aggregation in einem bestimmten Zustand.
2. Bei Sequenzdiagrammen kann auf Methoden bzw. Attribute des statischen Objektmodells Bezug genommen werden.

e) Zusammenhang zwischen den Modellen

1. Jedes Zustandsdiagramm ist umkehrbar eindeutig mit einem Objekt bzw. einer Klasse verbunden.
2. Das dynamische Verhalten (Zustandsdiagramm) einer Klasse wird an ihre Unterklassen vererbt.
3. Ereignisse aus der Modellierung des dynamischen Objektverhaltens können Methoden der jeweiligen Klasse zugeordnet werden.
4. Bei der Modellierung von Zuständen im dynamischen Modell kann auf Attribute des Objektmodells Bezug genommen werden.

f) Abstraktionsebenen / Sichten

1. Der Detaillierungsgrad von Klassendiagrammen am Bildschirm läßt sich durch Ausblenden bzw. Einblenden von Attributen und Methoden individuell einstellen.
2. Die Darstellung von Attributen und Methoden ist auf unterschiedlichem Detaillierungsniveau möglich (Ein-/Ausblenden von Typen, Rückgabetypen, Parameter, Parametertypen etc.).
3. Die Angaben über die Sichtbarkeit von Objektkomponenten können für eine Variation der Genauigkeit der Darstellung herangezogen werden (z.B. lediglich Anzeige der öffentlichen Komponenten).
4. Verfeinerte Zustandsdiagramme können in verschiedenen Detaillierungsstufen angezeigt werden.
5. Das Detaillierungsniveau der Diagrammanzeige kann sowohl für jedes Diagramm einzeln, als auch generell als Default-Einstellung festgelegt werden.

g) Vorgangsreihenfolge bei der Entwicklung

1. Das Werkzeug zwingt nicht zu einer bestimmten Vorgangsweise, was die Reihenfolge des Anlegens von Modellierungsinformation betrifft. Klassen etwa, können zunächst nur mit ihrem Namen angelegt und erst später genauer beschrieben werden.
2. Modellierungselemente können direkt textuell im Data-Dictionary erfaßt und erst später in ein oder mehrere Diagramme eingezeichnet werden.
3. Neu anzulegende Modellierungselemente können direkt in ein Diagramm eingezeichnet werden, wobei die entstehenden Informationen automatisch sofort im Data-Dictionary erfaßt werden.
4. Bestimmte Abläufe, welche die Einhaltung eines Vorgehensmodells gewährleisten sollen, können im Tool eingegeben werden. Diese Ablaufschritte, die dann vom Tool vorgegeben werden, müssen aber nicht eingehalten werden, sondern können auch übersprungen werden (*Phasenmodellierung*).

h) Unterstützung bei der Methodenanwendung

1. Zur Unterstützung bei der Anwendung von UML lassen sich aus dem Tool Informationen zu deren Notation und Semantik in Textform abrufen (und ausdrucken).
2. Es gibt eine hypertextartige "Methodenhilfe", welche die Elemente von UML übersichtlich beschreibt.
3. Es gibt eine Art "Methoden-Tutorial", welches die Anwendung von UML in einem Musterbeispiel veranschaulicht.
4. Die Notationselemente des Tools entsprechen 1:1 jenen der UML, sodaß die Verwendung von Basisliteratur zur Methode keine Verwirrung auslöst.
5. Bei der Eingabe von Modellierungsinformationen, die der Konsistenzprüfung bzw. den Integritätsbedingungen nicht standhalten, er-

scheint eine Warnung und eine Erklärung der entsprechenden Konzepte ist abrufbar.

6. Neben der Tool-Dokumentation gibt es ein Handbuch, das UML beschreibt.

4.2 Funktionale Kriterien

a) Datenhaltung

1. Das Tool verfügt über eine Datenbasis, welche die Informationen für alle Teilmodelle zur Verfügung stellt (Data-Dictionary, Repository, Enzyklopädie etc.).
2. Modellinformationen, die in mehreren Diagrammen vorkommen, sind in der Datenbasis nicht redundant bzw. nur einmal gespeichert (*Integration*).
3. Modellinformationen, die in der Datenbasis erfaßt sind, müssen nicht zwingend in einem Diagramm vorkommen.
4. Modellinformationen können auch direkt in der Datenbasis gewartet (insbesondere auch gelöscht) werden.
5. Das Datenformat kann als "offen" bezeichnet werden und ist nicht ausschließlich in diesem Tool anzutreffen.
6. Das Tool baut auf einem OODBS (Objektorientierten Datenbanksystem) auf.
7. Es gibt eine ASCII-Schnittstelle für Datenimport und -export in, bzw. aus der Datenbasis.
8. Der Import und Export von Daten in die bzw. aus der Datenbasis ist über Schnittstellen zu relationalen Datenbanken möglich.

b) Browsing-Eigenschaften

1. Das Auffinden von beliebigen Elementen (Klassen, Attribute, Methoden etc.) durch Eingabe ihres Namens ist möglich.
2. Elemente können auch nach Ähnlichkeitskriterien, durch *pattern matching*, gesucht werden.
3. Die Suche von Elementen kann durch bloße Eingabe der Anfangsbuchstaben vorgenommen werden, wodurch die jeweils zutreffenden Elemente zur Auswahl angeboten werden.
4. Bei der Suche können Einschränkungen vorgenommen werden, etwa durch das Setzen von Filtern.
5. Als Ergebnis einer Suche erhält man das gesuchte Element mit weiteren wichtigen Informationen - z.B. für eine Klasse: Oberklasse, Unterklassen, Beziehungen zu weiteren Klassen, Attribute, Methoden.

6. Die Angaben, die zu einem Modellierungselement gemacht werden, dienen unmittelbar als *Link* zum Zugriff auf genauere Informationen über diese selbst (z.B. Eigenschaften von Methodenparametern).
7. Das Werkzeug ermöglicht die Durchführung von Änderungen an beliebigen Modellierungsinformationen direkt aus dem Browser heraus. Diagramme, auf die solche Änderungen Auswirkungen haben, werden sofort automatisch angepaßt.
8. Das Werkzeug ermöglicht auch in textueller Form eine übersichtliche Darstellung von Klassenhierarchien.
9. Das Tool ermöglicht auf einfache Art die Navigation innerhalb einer Klassenhierarchie (z.B. durch einfaches Anklicken der Oberklasse).
10. Es gibt die Möglichkeit, für jedes Modellierungselement eine Aufstellung zu erhalten, in welchen übergeordneten Strukturen es vorkommt (z.B. In welchen Klassen kommt das Attribut "GebDatum" vor?).
11. Alle Modellierungselemente, die in der Datenbasis gespeichert sind, aber noch in keinem Diagramm vorkommen, können auf einfache Weise ausgewählt werden.

c) Navigation im Systemmodell

1. Das Werkzeug erlaubt eine rasche und übersichtliche Navigation innerhalb bzw. zwischen den Darstellungen für einzelne Modellierungsaspekte eines Systems.
2. Es gibt die Möglichkeit, ausgehend von einem Diagramm direkt zur jeweils übergeordneten bzw. verfeinernden Darstellung zu wechseln. Z.B. kann durch einfachen Mausklick aus einem Zustandsdiagramm zum entsprechenden Objekt- bzw. Klassendiagramm verzweigt werden.
3. Zusammenhänge bzw. Verbindungen (*Links*) zur Navigation zwischen verschiedenen Diagrammen können vom Benutzer beliebig angelegt und manipuliert werden.
4. Die einzelnen Elemente eines Diagramms sind nicht nur textuelle bzw. graphische Darstellungen sondern dienen gleichzeitig zum Aufrufen von elementspezifischen, genaueren Informationen (z.B. Öffnen des Eigenschaftendialogs durch Anklicken des jeweiligen Elements).

d) Konsistenzprüfung / Integrität

1. Es lassen sich nur syntaktisch richtige Diagramme anlegen. Bereits während der Modellierung werden ungültige Konstrukte verhindert.
2. Wird ein Notationselement angelegt, so muß dieses auch sofort benannt werden. Es gibt keine "namenlosen" Elemente
3. Es können keine Assoziationen, Datenflüsse oder Transitionen angelegt werden, die ins "Leere" gehen.
4. Rekursive Vererbungsstrukturen können nicht erstellt werden.
5. Mehrere Modellierungselemente mit gleichem Namen können nicht angelegt werden.

6. Bei der Anlage von Namen, die bestehendenden ähnlich sind, wird der Benutzer gewarnt, z.B. "GebDatum" und "Gebdatum".
7. Das Tool verfügt über eine Funktion, mit der ein bestehendes Modell bzw. der gesamte Datenbestand in der Datenbasis auf Konsistenz geprüft werden kann. Dabei werden auch Beziehungen zwischen den Teilmodellen bzw. zwischen Diagrammen geprüft.
8. Die Anzeige aller angelegten, aber nie verwendeten Modellierungselemente ist möglich.
9. Wird ein Modellierungselement aus einem Diagramm gelöscht, darf es nicht gleich automatisch aus der Datenbasis gelöscht werden (*referentielle Integrität*).

e) Report-Ausgabe

1. Basierend auf den Daten des Data-Dictionary lassen sich automatisch Auswertungen mit einstellbarem Detaillierungsgrad ausgeben.
2. Für die statische Objektstruktur (Klassen, Attribute, Methoden etc.) können geeignete Reports ausgegeben werden.
3. Für das dynamische Objektverhalten lassen sich geeignete Reports ausgeben.
4. Für die statische Systemstruktur (Vererbungs-, Aggregations-, Assoziationsbeziehungen) lassen sich geeignete Reports erstellen.
5. Auch zur Darstellung des dynamischen Systemverhaltens (z.B. Inhalt der Sequenzdiagramme) gibt es Reports.
6. Ein Report, der über die Ergebnisse einer expliziten Konsistenzprüfung Informationen enthält, läßt sich generieren.
7. Reports können auch einfach auf Grund von ad-hoc Abfragen oder auf andere, frei definierbare Weise erstellt werden.
8. Die Ausgabe von Reports im PostScript-Format ist möglich.
9. Die Ausgabe von Reports im ASCII-Format ist möglich.
10. Die Ausgabe von Reports im Format gängiger Text- bzw. Dokumentenverarbeitungssysteme ist vorgesehen.

f) Datenbankschema-Ausgabe

1. Die Generierung eines Datenbankschemas für relationale Datenbanken auf Basis der Informationen des Objektmodells ist möglich.
2. Für den Fall der Verwendung von relationalen Datenbanken ist eine automatische Erstellung eines erweiterten ER-Diagramms (EERD) aus den Informationen des Data-Dictionary möglich.
3. Die Generierung eines objektorientierten Datenbankschemas auf Basis der Informationen des Objektmodells ist möglich.
4. Die Angabe von Schlüsselattributen für eine spätere DB-Schema-Ausgabe ist möglich.
5. Zusätzlich zu den Datentypen für die bei der Codegenerierung zu verwendende Zielsprache können bei den Attributen auch Domains für Datenbanken angegeben werden.

6. Alle datenbankrelevanten Informationen aus der statischen Klassenstruktur und der statischen Struktur des Anwendungssystems finden sich in den generierten Datenbankschemata wieder (Attribute, Beziehungen, Vielfachheiten etc.).

g) Diagramm-Ausgabe

1. Bei der Diagramm-Ausgabe ergeben sich keine Probleme mit der Unterstützung von gängigen Druckertypen. Der Diagrammausdruck kann als präsentationsfähig bezeichnet werden.
2. Alle Diagramme lassen sich so ausdrucken, daß sie durch einfaches Zusammenkleben von A4 Blättern übersichtliche Dokumentationen ergeben (kachelartiger Ausdruck).
3. Bei der Diagramm-Ausgabe ist das Verhältnis zwischen Qualität und Schnelligkeit des Ausdrucks einstellbar (vom Tool unterstützt).
4. Die gedruckten Diagramme entsprechen genau der Darstellung am Bildschirm.
5. Das Abstraktionsniveau läßt sich für die Diagrammausgabe frei wählen (z.B. nur Klassennamen anzeigen / Klassen, Attribute u. Methoden anzeigen etc.).
6. Bei der Diagramm-Ausgabe sind die Ausrichtung (Quer- oder Hochformat) und Skalierung der Darstellung einstellbar.

h) Offenheit - Import / Export

1. Das Tool ist ein Modul aus der Produktpalette des Herstellers, welches mit Zusatzprodukten des selben Herstellers zusammenarbeitet und auf diese Weise modulartig erweitert werden kann.
2. Das Tool schöpft die von der Plattform gebotenen Möglichkeiten des Datenaustauschs sowohl für Text als auch für graphische Darstellungen aus (z.B. Cut&Paste von Text und Graphik über die Zwischenablage).
3. Das Werkzeug unterstützt die Konzepte einer Plattform für objekt- bzw. dokumentenzentriertes Arbeiten (z.B. OLE, SOM).
4. Die Informationen des Data-Dictionaries lassen sich auch als Code eines Hypertext-Formats ausgeben (SGML-Ausgabe, z.B. HTML).
5. Das Tool arbeitet optimal mit Standardprodukten anderer Hersteller zusammen und läßt sich auf diese Weise gut in eine bestehende Entwicklungsumgebung einbinden (z.B. Unterstützung von Textverarbeitungs- oder Projektmanagement-Software etc.).
6. In den Graphikeditor können auch geläufige toolfremde Grafikformate eingelesen werden.
7. In den bzw. die Zeicheneditoren können auch geläufige Textformate direkt eingelesen werden.
8. Es gibt bereits Zusatztools von Drittherstellern, die Ausgaben des Tools einlesen und weiterverarbeiten können (z.B. DTP-Systeme, Hypertext-Generatoren etc.).

i) Code-Generierung

1. Die automatische Codegenerierung für eine oder mehrere in Frage kommende objektorientierte Zielsprachen ist möglich.
2. Für jede im Modell vorhandene Klasse wird im Sourcecode der Zielsprache eine entsprechenden Codierung vorgenommen.
3. Attribute werden inklusive ihrer Typen in die Zielsprache umgesetzt.
4. Methoden werden inklusive ihrer Argumente (inkl. Typen) und Rückgabetypen in den entsprechenden Sourcecode umgesetzt.
5. Beziehungen zwischen Klassen werden im resultierenden Sourcecode richtig umgesetzt.
6. Für Methoden werden Sourcecode-Gerüste produziert, die noch durch den Inhalt ergänzt werden müssen (*body*).
7. Zugriffsbeschränkungen werden auch im Programmcode berücksichtigt.
8. Strukturierungen oberhalb des Niveaus der Klassen finden sich in der physischen Struktur des Codes wieder (z.B. mehrere Module führen zu einzelnen Sourcecode-Dateien).
9. Das Ausmaß der Codegenerierung läßt sich durch Parameter bestimmen (z.B. Defaultwerte, Datentypen, Rückgabewerte, Argumente, Zugriffsberechtigungen, Präfix für Klassennamen etc.).
10. Wenn generierter Code händisch erweitert wurde und die automatische Codegenerierung erneut angestoßen wird, bleibt der eigene (*user added*) Code erhalten (z.B. Kennzeichnung durch toolseitig vergebene Kommentare).
11. Inhalte von Methoden lassen sich - fragmentartig - schon in der textuellen Beschreibung der Methode eintragen (Pseudocode bzw. bereits syntaktisch richtig) und werden in den Code übernommen.

j) Reverse-Engineering

1. Bestehender Programmcode kann mit dem Tool eingelesen werden. Dieses erzeugt aus den Code-Informationen die entsprechenden A/D-Diagramme.
2. Es kann auch Code eingelesen werden, der nicht mit dem Tool selbst generiert wurde.
3. Codefragmente, die über Klassendefinitionen, Komponenten und Beziehungen hinausgehen (z.B. der Inhalt von Methoden - *body*) können in einem Editor bearbeitet und erneut als Code gespeichert werden.

k) Test / Verifikation

1. Das Tool erlaubt die textuelle Beschreibung von Testfällen für einzelne Objekte bzw. Methoden.
2. Das Tool ermöglicht eine Simulation auf Modellebene, bei der das Senden von Nachrichten simuliert und das entsprechende Objektverhalten animiert dargestellt wird.

l) Maßzahlen und Statistiken

1. Das Tool kann auf Basis der Daten des Repository geeignete Software-Maßzahlen berechnen, etwa die Tiefe von Vererbungshierarchien oder Kopplungsmaßzahlen etc.
2. Die automatische Ausgabe von Statistiken (Anzahl modellierter Klassen, Attribute, Methoden etc.) ist möglich.
3. Die Ausgabe von Statistiken, die für das Projektmanagement interessant sind, ist möglich (z.B. Anzahl der wiederverwendeten Klassen aus anderen Projekten).
4. Die Ausgabe von Informationen über den Projektstatus (Mitarbeiter, Zugriffsberechtigungen, letzte Änderungen etc.) ist möglich.

m) Multi-User - Eigenschaften

1. Das OO-CASE-Tool ist in einer Netzwerkumgebung (LAN) lauffähig und kann von mehreren Benutzern gleichzeitig verwendet werden.
2. Das Tool kann auf Multi-User-Plattformen eingesetzt werden (UNIX, OS/2), auch in heterogenen Umgebungen.
3. Die Benutzer des Werkzeugs können - unabhängig vom Basis-Betriebssystem - in User und Gruppen eingeteilt werden (tooleigenes "Userkonzept").
4. Es gibt einen Konfliktlösungsmechanismus, der verhindert, daß kein Benutzer die Arbeit eines anderen versehentlich zerstört (*Locking*).
5. Der Umfang des *Locking* ist einstellbar - z.B. von Modellebene bis zur Ebene eines Attributs.
6. Der Versand von Nachrichten (E-Mail) an einen oder mehrere andere Benutzer ist direkt aus dem Tool heraus möglich.
7. Das System bietet eine Funktion zum interaktiven Dialog mit einem anderen Benutzer (*talk*-Funktion).

n) Security

1. Jeder Benutzer muß sich mit seinem User-Namen und einem Paßwort ausweisen, bevor er das Werkzeug nutzen kann (*login/logout*).
2. Zugriffsrechte für Benutzer können auf User- oder Gruppenebene festgelegt werden.
3. Zusätzlich können Rechte auf Phasenebene festgelegt werden (z.B. User A darf keinen Code generieren).
4. Das Anlegen von Usern und Gruppen und die Rechteverwaltung können nur von einem ausgezeichneten Benutzer (Systemadministrator) durchgeführt werden.
5. Über ein automatisch mitgeschriebenes Protokoll können Änderungen auch im nachhinein auf den dafür verantwortlichen Benutzer zurückgeführt werden (*Logging*).
6. Das Tool verfügt über eine geeignete Möglichkeit zur Datensicherung, die auch automatisierbar ist.

7. Für den Fall von Systemabstürzen verfügt das Werkzeug über einen geeigneten Recovery-Mechanismus, der die Integrität des Programms und der Daten gewährleistet.
8. Jeder Benutzer kann Elemente eines Modells vorübergehend vor Änderungen schützen (*einfrieren*), auch falls er nicht gerade daran arbeitet und eingeloggt ist.
9. Im Falle von Bedienungsfehlern verbleibt das Werkzeug in einem definierten Zustand und gibt eine Fehlermeldung aus.
10. Unbeabsichtigte Löschungen können durch "Bestätigungsabfragen" verhindert werden.
11. Löschungen können bis zu einem gewissen Grad automatisch rückgängig gemacht werden (generelle *Undo*-Funktion).

o) Multi-Projekt-Eigenschaften / Versionsmanagement

1. Das Werkzeug ermöglicht die zeitlich parallele Abwicklung von mehreren Projekten.
2. Der Austausch von Modellierungsinformationen durch Kopieren zwischen einzelnen Projektdatenbasen ist möglich.
3. Der Bezug auf bereits in anderen Projekten gespeicherte Modellierungsinformationen (*Referenzen*) ist möglich. Somit können bereits erstellte Modelle wiederverwendet werden, ohne diese kopieren zu müssen.
4. Das Werkzeug erlaubt die gleichzeitige Verwaltung von mehreren verschiedenen Versionen eines Systemmodells.
5. Das Tool kann automatisch Änderungen einer Version gegenüber der Ausgangsversion anzeigen.

4.3 Ergonomie

a) Benutzeroberfläche

1. Das allgemeine Erscheinungsbild ist optisch ansprechend und übersichtlich.
2. Das GUI ist so implementiert, daß das Look-and-Feel der entsprechenden Benutzeroberfläche vorhanden ist.
3. Dialoge und Eingabemasken sind übersichtlich und zweckmäßig zu bedienen.
4. Die Namensgebung von Menüaktionen und Auswahlmöglichkeiten sind weitgehend selbstbeschreibend, einheitlich und widerspruchsfrei.
5. Momentan nicht verfügbare Menübefehle oder Optionen sind besonders dargestellt.

6. Im Falle von länger dauernden Aktionen erhält der Benutzer Informationen über Status und Verarbeitungsfortschritt (z.B. Fortschritts-Balken, Sanduhr).
7. Die Anzeige von Listen am Bildschirm ist einheitlich und übersichtlich.

b) Tool-Dokumentation

1. Es gibt ein ausführliches, gedrucktes Handbuch zur Bedienung des Tools.
2. Es gibt schriftliche Anleitungen zur Installation.
3. Das Handbuch ist in der jeweiligen Landessprache verfaßt.
4. Es gibt ein Kurz-Handbuch für versierte Benutzer.
5. Es gibt ein "Erste Schritte"-Handbuch für unerfahrene Benutzer.
6. Das Handbuch hat ein Inhaltsverzeichnis und einen Stichwortindex bzw. Referenzteil.
7. Das Handbuch läßt sich leicht ergänzen, Teile lassen sich leicht austauschen.
8. Es gibt ein hypertextartiges On-Line Handbuch.
9. Das On-Line Handbuch ist direkt aus dem Tool heraus abrufbar.
10. Das On-Line Handbuch kann im Hintergrund offen gehalten werden.
11. Mit dem Tool werden fertige Beispiele geliefert, die als Anregung und zur Einführung verwendet werden können, und die sich zum Selbststudium eignen.
12. Wichtige Informationen, die sich erst nach Vorliegen der Dokumentation ergeben haben, sind in Textdateien vorhanden, auf die auch die Dokumentation verweist.

c) Usability

1. Der Abbruch von Aktionen ist zu jeder Zeit möglich.
2. Im Fehlerfall wird der Benutzer durch eine informative und weiterführende Fehlermeldung benachrichtigt. Er erhält auf Wunsch auch direkt Hilfe zur aktuellen Situation
3. Für wichtige Menüfunktionen gibt es Abkürzungs-Tastenkombinationen (*Tastatur-Shortcuts*).
4. Funktionstasten können mit oft benötigten Befehlsfolgen belegt werden.
5. Das Tool bietet für ein rasches Arbeiten Kontextmenüs mit den jeweils anwendbaren, wichtigsten Befehlen an.
6. Die Eingabe von ungültigen Werten bzw. Ausführung ungültiger Operationen führt zu einer kontrollierten Fehlerbehandlung: Es erscheint eine Fehlermeldung und der kontrollierte Programmablauf wird fortgesetzt.
7. Es gibt Symbolleisten (*Toolbar*, *Buttonbar*), aus denen man häufig benötigte Befehle abrufen kann - etwa eine Werkzeugleiste mit Sym-

bolen zur Objektmodellierung. Die Symbolleisten sind vom Benutzer individuell konfigurierbar.

8. In einer Statuszeile finden sich hilfreiche Informationen - etwa über den gerade auszuführenden Befehl und die weitere Vorgangsweise.
9. Mehrfenstertechnik wird unterstützt (z.B. Windows MDI) d.h. mehrere Darstellungen desselben Projekts, z.B. verschiedene Diagramme können gleichzeitig nebeneinander betrachtet werden.
10. Die Selektion aus Listen ist auf einfache Weise möglich (Einfach- und Mehrfachauswahl).
11. Es gibt eine Version des Werkzeugs in der jeweiligen Landessprache.

d) Hilfefunktionen

1. Das Tool bietet eine kontextsensitive Hilfefunktion, die leicht erreichbar ist (z.B. *F1*).
2. Das Werkzeug hat eine Hypertext-Hilfe, welche alle wichtigen Hilfethemen umfaßt.
3. Die Tool-Hilfe ermöglicht das Suchen nach bestimmten Hilfethemen.
4. Das Werkzeug bietet eine *Hot-Help* an: Wird die Maus auf ein Element bzw. einen Menübefehl bewegt, erscheint dazu auf Wunsch Hilfe.

e) Graphikeditor / Darstellung

1. Die Symbole sind in einem bestimmten Umfang standardmäßig anpaßbar (als Voreinstellung).
2. Zusätzlich zu den Diagrammen aus UML können "freie" Diagramme aus den gängigsten Umrißformen (Rechtecke, Ovale, Linien etc.) mit Beschriftungen erstellt werden.
3. Die Darstellung der Diagramme am Bildschirm entspricht dem späteren Ausdruck (*WYSIWIG*).
4. Für Notationselemente können verschiedene, selbst definierbare Farben voreingestellt werden.
5. Es gibt eine Funktion zur horizontalen bzw. vertikalen Ausrichtung von Text oder Symbolen (*Alignment*).
6. Die Bildschirmdarstellung ganzer Diagramme oder Teile eines Diagramms kann (stufenlos) vergrößert und verkleinert werden (*Zoom*).
7. Es gibt eine Funktion zum Rotieren von Symbolen und vor allem von Text.
8. Es existiert eine Funktion, die ein Diagramm automatisch in der optimalen Größe und ohne Überlappungen von Elementen am Schirm anzeigt.
9. Das Layout eines Diagramms, welches man speichert und wieder lädt, wird beibehalten.
10. Die Schriftgröße zur Beschreibung von Diagrammelementen ist frei wählbar und die Schrift selbst gut lesbar.

11. Die Größe aller Notationselemente ist einstellbar, aber insgesamt einheitlich und nicht etwa von der Länge der darin enthaltenen Beschreibung (z.B. Klassenname) abhängig.
12. Ein Zeilenumbruch innerhalb einer Element-Beschriftung ist möglich.
13. Für alle Linien (z.B. bei Assoziationen) können auch nachträglich beliebige Eckpunkte festgelegt werden, um Überschneidungen zu vermeiden.
14. Der Editor erlaubt die Einstellung von Strichstärke und Linienart.
15. Es gibt eine Funktion zum "Neuzeichnen" des Bildschirminhalts durch das System (*Refresh*).

f) Graphikeditor / Bedienung

1. Das Einzeichnen von Notationselementen in ein Diagramm erfolgt komfortabel durch einfaches Aufziehen mit der Maus oder durch Auswahl aus einem Kontextmenü.
2. Sofort nach dem Anlegen eines Notationselements kann dieses textuell beschrieben werden (z.B. Klassenname, Attribut- und Methodennamen).
3. Eine Funktion zum vielfachen Einfügen desselben Elements hintereinander steht zur Verfügung (*Tool halten*).
4. Notationselemente können durch einfaches Drag&Drop an eine beliebige Stelle eines Diagramms verschoben werden.
5. Der Zeichenbereich für Diagramme ist "unendlich" groß. Werden Diagramme größer als eine Bildschirmseite, kann horizontal und vertikal gescrollt werden.
6. Zum Zweck einer automatischen, horizontalen bzw. vertikalen Ausrichtung von Diagrammelementen kann ein frei definierbares Raster angelegt werden (*Grid Alignment*).
7. Auf Wunsch erscheinen Linien oder Symbole exakt dort, wo man sie haben will.
8. Man kann mehrere Symbole gruppieren (z.B. durch Umrahmen mit dem Mauscursor), um darauf eine Operation auszuführen (z.B. Verschieben).
9. Das Abbrechen einer Zeichenaktion (z.B. irrtümliches Aufziehen einer Assoziation zu einer noch nicht gezeichneten, zweiten Klasse) ist einfach durchführbar.
10. Es gibt eine *Undo*-Funktion, die den letzten Bearbeitungsschritt (oder mehrere) rückgängig macht.
11. Es gibt eine *Redo*-Funktion, die den letzten Bearbeitungsschritt wiederholt.
12. Die allgemeine Bedienung des Graphikeditors verdient das Prädikat "intuitiv".
13. Im Graphikeditor wird das Kopieren/Ausschneiden und Einfügen von Symbolen unterstützt (*Copy/Cut & Paste*).
14. Symbole lassen sich leicht wieder aus einem Diagramm entfernen, etwa durch Markieren und Betätigen der *Entf*-Taste.

g) Performance

1. Bei der Darstellung von Diagrammen und deren Manipulation (Zoomen, Scrollen etc.) erfolgt der Bildaufbau in Anbetracht der zur Verfügung stehenden Hardwareumgebung ausreichend schnell.
2. Die Zeit, die das Tool zum Laden und Speichern von Projekten, Diagrammen oder Spezifikationen benötigt, erlaubt ein zügiges Arbeiten.
3. Suchergebnisse stehen, in Anbetracht des bereits erfaßten Datenvolumens, rasch zur Verfügung.
4. Die automatische Codegenerierung erfolgt ausreichend schnell.

h) Adaptivität

1. Das Werkzeug kann dem Lernfortschritt der Anwender angepaßt werden, z.B. Zurverfügungstellung unterschiedlich detaillierter Menüs (Ausblenden von komplexen Befehlen), Ein- und Ausschalten von Sicherheitsabfragen und Hinweisdialogen, *Expert* oder *Apprentice*-Modus etc.
2. Die Oberfläche läßt sich individuell gestalten (Symbolleisten, Funktionstastenbelegung etc.) und kann für jeden Benutzer spezifisch eingestellt werden.
3. Die Struktur der Menüs kann vom Anwender verändert werden (neue Befehle einfügen, Befehle umbenennen etc.).
4. Es gibt eine Möglichkeit, benutzerdefinierbare Befehlsfolgen aufzuzeichnen und später abzurufen (Makros).
5. Zur Erstellung von Dokumenten gibt es eine Script-Sprache, mit der Ausgaben frei gestaltbar sind. Alle Modellierungselemente können dabei als *Variablen* verwendet werden.
6. Das Werkzeug wird im Sourcecode ausgeliefert, was ein Maximum an Anpassungsmöglichkeiten bietet.

5 Evaluationen

Die Ergebnisse der Auswertungen beziehen sich auf den in Kapitel 4 vorgestellten Kriterienkatalog. Jeder Aspekt wird einerseits allgemein beschrieben, zusätzlich findet sich in der Dokumentmarginale eine tabellarische Bewertung der einzelnen Unterpunkte aus dem Katalog, gemäß folgendem Schema:

+	Funktion ist gut umgesetzt, bzw. vorhanden
0	Funktion ist mittelmäßig umgesetzt, bzw. teilweise vorhanden
-	Funktion ist schlecht umgesetzt, bzw. nicht vorhanden

5.1 Cool:Jex 4.1

Das CASE-Werkzeug Cool:Jex der texanischen Firma Sterling Software basiert auf dem Programm *ObjectTeam* der Firma Cayenne Software aus Massachusetts, die von Sterling gekauft wurde. Projektdaten werden in einem Repository gespeichert. Das Repository speichert die Daten des Systems teilweise in einer Datenbank und teilweise im Dateisystem. Die eingesetzte Datenbank hängt vom Betriebssystem ab, unter dem das Programm installiert ist. Unter Windows NT und Windows 9x kommt SQL Anywhere zum Einsatz, auf UNIX-Plattformen entweder Informix oder Oracle.

Cool:Jex ist ein sehr mächtiges Werkzeug, das sich über veränder- und erweiterbare Module an die Bedürfnisse des Benutzers anpassen läßt. Andererseits ist das Programm keineswegs einfach und intuitiv zu bedienen. Die Benutzeroberfläche wirkt etwas veraltet einige Funktionen sind nicht so zu erreichen, wie man von einem Windows-Programm erwarten würde. Dies wäre zum Beispiel die Entfernen-Taste zum Löschen von Elementen, oder ein Klick auf einen angewählten Knoten im Browser um dessen Namen zu verändern. Die Leistungsfähigkeit von Cool:Jex wird nur ein erfahrener und im Umgang mit dem Werkzeug geübter User ausnutzen können.

5.1.1 Methodenbasierte Kriterien

a) Modellierung der Objektstruktur - Objektstatik

Das Werkzeug speichert alle Daten des Modells in einem Repository. Der Inhalt des Repositorys kann über den Browser eingesehen werden, dies jedoch mit Einschränkungen: es gibt zum Beispiel keine Möglichkeit, alle im Repository vorhandenen Elemente eines Typs anzuzeigen (z.B. Klassen). Der

Browser zeigt nur die Notationselemente an, die das gerade gewählte Pakkage enthält.

Die Eigenschaften von Attributen und Operationen einer Klasse werden direkt beim Eintragen des Textes in das Klassendiagramm angegeben. Eingabemasken, die den Benutzer bei dieser Arbeit unterstützen sind nicht verfügbar.

Für Attribute kann neben Datentyp, Sichtbarkeit, Änderbarkeit und Lebensdauer auch festgelegt werden, ob das Attribut im Primärschlüssel vorkommen muß. Dies gibt im Allgemeinen nur Sinn, wenn die Zielsprache persistente Objekte unterstützt.

Für Methoden kann zusätzlich eine Parameterliste und ein Rückgabetyp angegeben werden. Die Parameter sind wiederum durch einen Typ und eine Parameter-Art (*IN*, *OUT*, *INOUT*) charakterisiert. Weiters kann eine Methode als abstrakt, oder als reine Zugriffsoperation definiert werden.

Die Reihenfolge der Komponenten einer Klasse kann nicht verändert werden und Cool:Jex läßt auch keine Angabe von Vor- oder Nachbedingungen zu.

Modellierung der Objektstruktur - Objektstatik, S. 38

1.	+
2.	+
3.	-
4.	+
5.	+
6.	+
7.	-

b) Modellierung der Systemstruktur - Systemstatik

Cool:Jex ist in der Lage, alle von der UML vorgegebenen Arten von Relationen darzustellen. Bereits erstellte Relationen können nachträglich verändert werden. Dies gilt sowohl für deren Eigenschaften, als auch für die beteiligten Klassen (Beziehungen können "umgehängt" werden). Abgesehen von Paketen ist keine Einteilung von Klassen zu Gruppen möglich. OCL-Notationen werden nicht unterstützt.

Modellierung der Systemstruktur - Systemstatik, S. 39

1.	+
2.	+
3.	-
4.	-

c) Dynamisches Objektverhalten - Objektdynamik

Für jede Klasse im System kann ein Zustandsdiagramm erstellt werden, welches eindeutig dieser Klasse zugeordnet ist. Dieses Diagramm kann über den Browser, jedoch auch durch einen Doppelklick auf das betreffende Klassensymbol im Klassendiagramm geöffnet werden.

Den Zustandsübergängen sind Ereignisse, Bedingungen und Aktionen zugeordnet. Hierbei kann auf die Attribute und Methoden der Klasse Bezug genommen werden. Jedem Modellierungselement im Zustandsdiagramm kann ein Text zur genaueren Beschreibung beigefügt werden.

Das Werkzeug ist in der Lage, geschachtelte Zustandsdiagramme (Überzustand, Unterzustände), History-Zustände und komplexe Zustandsübergänge darzustellen.

Cool:Jex stellt horizontale und vertikale *Concurrent State Separators* bereit, mit denen ein Superzustand in mehrere nebenläufige Zustandsregionen unterteilt werden kann.

Dynamisches Objektverhalten - Objektdynamik, S. 39

1.	+
2.	+
3.	+
4.	+
5.	+
6.	-

d) Dynamisches Systemverhalten - Systemdynamik

Wie alle anderen Notationselemente werden auch die Eigenschaften von Nachrichten als String direkt in das Sequenzdiagramm eingetragen. Dabei kann eine Bedingung sowie mehrere Parameter angegeben werden.

Es kann sowohl auf Methoden, als auch - bei Bedingungen - auf Attribute des statischen Objektmodells Bezug genommen werden. Außerdem erlaubt das Programm das Setzen von Zeitmarkern an Anfang und Ende von Nachrichten, sowie das Angeben von Zeiteinschränkungen, um Zeitintervalle zwischen ein oder mehreren Nachrichten zu dokumentieren.

Dynamisches Systemverhalten - Systemdynamik, S. 39

1.	+
2.	+

e) Zusammenhang zwischen den Modellen

Zustandsdiagramme sind eindeutig einer Klasse zugeordnet. Sie werden nicht an die Unterklassen ihrer zugehörigen Klasse weitervererbt. Weiters kann durch einen Doppelklick auf ein Modellelement in einem Diagramm zu verschiedenen anderen Diagrammen gewechselt werden, in denen das Element enthalten ist. Bei der Modellierung des dynamischen Modells kann sowohl auf Methoden, als auch auf Attribute des Statischen Modells Bezug genommen werden.

Zusammenhang zwischen den Modellen, S. 39

1.	+
2.	-
3.	+
4.	+

f) Abstraktionsebenen / Sichten

Der Detaillierungsgrad der Darstellung von Klassendiagrammen läßt sich in geringem Maße verändern. Zum Beispiel gestattet das Programm das Ein- und Ausblenden von Parametertypen, Sichtbarkeit und Standard-Zugriffsoperationen. Diese Einstellungen können sowohl für das gesamte Diagramm, als auch für jede Klasse separat vorgenommen werden. Wirklich umfangreiche Möglichkeiten, den Detaillierungsgrad der grafischen Darstellung zu manipulieren, wie zum Beispiel eine Veränderung der Detailtiefe abhängig von der Sichtbarkeit des Modellierungselementes, sind nicht verfügbar.

Die Einstellungen bezüglich der Darstellung im Diagrammeditor können einerseits für jedes Diagramm separat eingestellt werden, andererseits kann die Defaulteinstellung, über Systemvariable, verändert werden.

Abstraktionsebenen / Sichten, S. 40

1.	*0*
2.	*0*
3.	-
4.	-
5.	+

g) Vorgangsreihenfolge bei der Entwicklung

Cool:Jex zwingt den Benutzer nicht, ein Modellierungselement bei dessen Erstellung bis ins Detail zu beschreiben. Es ist ohne weiteres möglich, neue Elemente nur mit ihrem Namen zu beschreiben und deren Parameter zu einem späteren Zeitpunkt zu vervollständigen. Das Programm erlaubt sogar das Anlegen von unbeschrifteten Elementen, diese werden jedoch nicht in das Repository aufgenommen. Wird zum Beispiel eine namenlose Klasse im Klassendiagramm erstellt, wird nur die grafische Information in der Diagrammdatei abgelegt, wohingegen das Repository unberührt bleibt (bis die Klasse eventuell später benannt wird).

Das Programm erlaubt das Anlegen von Modellierungselementen nur im Diagrammeditor. Durch das Einzeichnen eines Elementes in ein Diagramm wird dieses automatisch in der zugrundeliegenden Datenbasis abgelegt. Der Browser zeigt den Inhalt des Repositorys an, gestattet jedoch nicht, die enthaltenen Elemente zu editieren oder zu löschen.

Cool:Jex beinhaltet ein Vorgehensmodell, das dem Benutzer eine Richtschnur für den Projektablauf vorgeben soll. Dieses Vorgehensmodell besteht aus folgenden vier Phasen:

- ❑ Analysephase
- ❑ Systemdesignphase
- ❑ Objektdesignphase
- ❑ Implementationsphase

Analysephase: Das Ziel der Analysephase ist, die Anforderungen für das Applikation zu definieren. Hierbei muß sichergestellt werden, daß sich Entwickler und Auftraggeber über die Funktionen und deren Umfang einig sind. Die wichtigsten Diagramme während der Analysephase sind Use-

Vorgangsreihenfolge bei der Entwicklung, S. 40

1.	*0*
2.	+
3.	+
4.	+

Case-, Sequenz- und Kollaborationsdiagramm zur Beschreibung bestimmter Szenarien, sowie das Klassendiagramm.

Systemdesignphase: Während der Systemdesignphase wird die Architektur der Applikation festgelegt, und diese in verschiedene Pakete aufgeteilt. Die optimale Architektur maximiert die Interaktionen zwischen Klassen eines Paketes und minimiert die Interaktionen zwischen Klassen verschiedener Pakete.

Objektdesignphase: In der Objektdesignphase werden die Klassendiagramme der verschiedenen Pakete verfeinert und verbessert, bis sie detailliert genug für die Implementation sind.

Implementationsphase: In dieser Phase wird der Code automatisch generiert und händisch erweitert.

Während der Arbeit an einem Projekt werden die erarbeiteten Diagramme von einer Phase zur nächsten weitergereicht und dort entsprechend weiterbearbeitet. Der Benutzer ist nicht an diese vier vorgegebenen Phasen gebunden. Es besteht die Möglichkeit, Phasen zu löschen beziehungsweise neue Phasen in das Modell einzufügen.

h) Unterstützung bei der Methodenanwendung

Unterstützung bei der Methodenanwendung, S. 40

1.	-
2.	-
3.	-
4.	+
5.	-
6.	-

Die Tooldokumentation geht kaum auf die UML selbst ein. Es wird vorausgesetzt, daß der Benutzer bereits mit dieser Notation vertraut ist, beziehungsweise, daß er allgemeine Informationen zu diesem Thema aus einer anderen Quelle bezieht. Die als HTML-Dokument vorliegende Online-Dokumentation geht zwar auf jede Diagrammart ein, das Hauptaugenmerk liegt hierbei allerdings bei der Beschreibung von Cool:Jex und nicht bei einer genauen Erklärung der UML.

Die Umsetzung der UML-Notation ist bei diesem Werkzeug sehr genau, und es werden mehr Feinheiten der Notation unterstützt als bei den meisten anderen Programmen dieser Art. Aus diesem Grund ist das Studium von Basisliteratur zur UML jedenfalls empfehlenswert.

Weitere Hilfestellung zur UML, wie ein Tutorial zu diesem Thema oder ein UML-Handbuch, bietet Cool:Jex nicht.

5.1.2 Funktionale Kriterien

a) Datenhaltung

Cool:Jex setzt auf dem sogenannten Sterling Repository auf, einer Datenbasis, in der die Projektdaten aller Projekte enthält. Diese Datenbasis ist als objektorientierte Datenbank ausgeführt. Welche Datenbank im speziellen zum Einsatz kommt, hängt vom verwendeten Betriebssystem ab. Unter Windows NT und Windows 9x kommt SQL Anywhere zum Einsatz, auf UNIX-Plattformen entweder Informix oder Oracle. Im Programmpaket sind außer diesem Repository auch Werkzeuge enthalten (Browser, Diagrammeditor,...), die mit den im Datenstamm enthaltenen Daten arbeiten und diese verändern können.

Mehrfach angelegte Modellelemente werden im Repository nur einmal gespeichert. Es ist zum Beispiel möglich, in einem Klassendiagramm zwei Klassen mit demselben Namen anzulegen, Cool:Jex erlaubt jedoch eine Manipulation der Eigenschaften dieser Klasse nur über eines der beiden grafischen Elemente (das zuerst angelegte), das andere ist lediglich grafischer Ballast. Weiters ist es möglich Klassen, die bereits in einem Paket vorkommen, in einem anderen Paket zu verwenden. Ruft der Benutzer den Eigenschaftendialog für dieses Modellelement vom zweiten Paket aus auf, gestattet ihm Cool:Jex nicht, die Daten zu verändern.

Datenhaltung, S. 41

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	-
8.	*0*

Werden Modellierungselemente aus einem Diagramm gelöscht, werden diese nicht aus der Datenbasis gelöscht. Generell ist es umständlich, Modellierungselemente vollständig aus dem Repository zu entfernen. Hierzu müssen zunächst alle Referenzen auf ein Element gelöscht werden. Jetzt kann es mit der Funktion *Delete Unreferenced Items* aus dem *Utilities*-Menü gelöscht werden.

Nur der Diagrammeditor erlaubt es, die Eigenschaften von Modellelementen zu bearbeiten. Der Repository-Browser ist dazu nicht in der Lage.

Da das dem System zugrundeliegende Repository, wie oben beschrieben, auf einem Datenbanksystem eines Drittanbieters basiert, ist hierdurch eine Schnittstelle zu anderen Datenbanksystemen vorhanden. Allerdings werden im Repository nur Informationen über die Modellelemente selbst, jedoch keine grafischen Informationen aus der Diagrammansicht gespeichert. Weiters besteht die Option, Modelle in ein Microsoft Repository zu exportieren bzw. daraus zu importieren. Auch mit dem UML-Werkzeug Rational Rose können Daten ausgetauscht werden. Über die Export- und Importmodule von Rose können Modellinformationen für Use-Case-, Klassen-, Sequenz-, und Kollaborationsdiagramme ausgetauscht werden. Auf UNIX-Systemen können Daten von Rose in Cool:Jex importiert werden, während der Datenaustausch auf Windows-Systemen beidseitig funktioniert. Eine Funktion, die die Datenbasis als ASCII-Datei exportiert, existieren nicht.

Als Extra wird mit Cool:Jex ein Repository-Management Werkzeug mitgeliefert, mit dem der Datenstamm gewartet werden kann. Unter anderem kann das Programm das Repository optimieren, sowie Backups einzelner Projekte oder des gesamten Repositorys anlegen. Die so entstehenden Bakkup-Dateien werden komprimiert und sind erfreulich klein. Die Backup-Datei eines 2 Mb großen Repositorys belegt gerade 10 kb.

b) Browsing-Eigenschaften

Um das Auffinden von Klassen, Attributen oder Methoden im System zu ermöglichen, beinhaltet das Programm einen sogenannten *Class Browser*. Dieser kann entweder auf Paketebene oder auf Phasenebene gestartet werden und zeigt dann alle Klassen des gewählten Paketes bzw. alle Klassen der aktuellen Phase. Im Fenster des Klassenbrowsers steht eine Liste aller Klassen zur Verfügung. Wird eine Klasse aus dieser Liste gewählt, stellt der Browser ihre Super- und Subklassen, ihre Attribute und Operationen sowie ihre Assoziationen dar. Der Klassenbrowser ist in der Lage, nach einer Klasse, einem Attribut oder einer Methode zu suchen, sowie zu einem Klassendiagramm, in dem eine bestimmte Klasse vorkommt, zu navigieren. Auch die Navigation zu einer Sub- bzw. Superklasse der gerade gewählten Klasse ist möglich.

Browsing-Eigenschaften, S. 41

1.	*0*
2.	-
3.	-
4.	-
5.	-
6.	-
7.	*0*
8.	*0*
9.	*0*
10.	+
11.	+

Es sind keine erweiterten Suchfunktionen wie *pattern matching* verfügbar. Suchfilter können für Attribute und Methoden spezifiziert werden, wobei Einschränkungen für Parameter wie Typ, Sichtbarkeit, Startwert, usw. getroffen werden können. Es ist auch nicht möglich, direkt vom Klassenbrowser aus in den Eigenschaftendialog eines ausgewählten Modellierungsele-

mentes zu wechseln. Abgesehen von den beschränkten Fähigkeiten des Klassenbrowsers, Klassenhierarchien darzustellen, gibt es auch einen Report, der diese in Textform ausgibt. Dieser Report beschränkt sich allerdings darauf, die Klassennamen anzugeben, wobei eventuelle Unterklassen einer Klasse eingerückt dargestellt werden.

Modellierungsinformationen, die in der Datenbasis gespeichert sind, aber von keinem Diagramm verwendet werden, können im Repository-Browser angezeigt werden.

c) Navigation im Systemmodell

Navigation im Systemmodell, S. 42

1. +
2. +
3. +
4. +

Cool:Jex gestattet eine schnelle und bequeme Navigation von Klassendiagrammen zu anderen, bestimmten Klassen zugeordneten Diagrammen. Zum Beispiel kann durch einen Doppelklick auf das Klassensymbol einer bestimmten Klasse im Klassendiagramm zu deren Zustandsdiagramm gesprungen werden, oder ein Zustandsdiagramm für die Klasse angelegt werden, falls noch keines existiert.

Cool:Jex stellt defaultmäßig eine Vielzahl solcher Verknüpfungen bereit, wie etwa:

- *Zustandsdiagramm*
  von einer Klasse zu einem zugehörigen Klassendiagramm
  von einem Zustand in ein weiteres Zustandsdiagramm oder ein Aktivitätsdiagramm
- *Use-Case Diagramm*
  von einem Aktor zu einem Klassen- oder Use-Case Diagramm
  von einem Use-Case zu einem Aktivitäts-, Kollaborations-, Sequenz- oder Use-Case Diagramm
- *Klassendiagramm*
  von einer Klasse zu Aktivitäts-, Kollaborations-, Sequenz-, Komponenten-, Verteilungs- oder Zustandsdiagramm, sowie zu einem verfeinerten Klassendiagramm.
- (usw.)

Die Navigationsbeziehungen können vom Benutzer nach eigenen Wünschen und Vorstellungen verändert werden. Hierzu müssen Änderungen in zwei *Customization Files* vorgenommen werden. In diesen Dateien sind *Open Strategies* festgelegt, die bestimmen, was passiert (was geöffnet werden soll), wenn der Benutzer eine Navigation von einem bestimmten Element aus in einem bestimmten Diagramm initiiert.

Eine weitere Navigationsmöglichkeit zwischen Diagrammen stellt der Grafikeditor bereit. Befindet sich der Benutzer im Diagrammeditor, kann er über vor/zurück Buttons zwischen den im Paket vorhandenen Diagrammen gleicher Art wechseln. Ist zum Beispiel der Editor mit einem Sequenzdiagramm geöffnet, kann sehr schnell zwischen allen Sequenzdiagrammen dieses Paketes gewechselt werden.

d) Konsistenzprüfung / Integrität

Cool:Jex besitzt keine Konsistenzprüfung, die schon während der Konstruktion des Diagramms dessen Richtigkeit sicherstellt. Es wird lediglich für die Einhaltung der rudimentärsten Regeln gesorgt, wie zum Beispiel, daß keine Relationen im Leeren enden. Unbenannte Klassen oder rekursive Generalisationen können erstellt werden, ohne daß das Programm eine Warnung

oder einen Fehler ausgibt. Erst ein vom Benutzer explizit ausgelöster Check spürt Fehler auf und gibt sie in einem Protokollfenster aus.

Das Werkzeug kennt vier verschiedene explizite Check-Kommandos:

Konsistenzprüfung / Integrität, S. 42

1.	-
2.	-
3.	+
4.	0
5.	+
6.	-
7.	+
8.	+
9.	+

- *Check Contents*
  Dieser Befehl überprüft die Semantik eines Diagramms. Zum Beispiel überprüft das Kommando bei einem Klassendiagramm, ob alle Link-Attribute mit einer Assoziation verbunden sind, bei einem Sequenzdiagramm, ob jedes Ereignis genau einen Initiator hat.
- *Check Local Model*
  Überprüft ausgewählte Klassen bzw. alle Klassen eines ausgewählten Diagrammes. Dieser Befehl überwacht zum Beispiel, daß keine Klasse ein Attribut enthält, das den gleichen Namen trägt wie ein an die Klasse geschicktes Ereignis.
- *Check Global Model*
  Diese Funktion entspricht *Check Local Mode*l, mit dem Unterschied, daß jetzt alle Klassen des gerade gewählten Paketes der Überprüfung unterzogen werden.
- *Check Use Case Model*
  Diese Funktion überprüft alle Use-Case Diagramme. Hier wird zum Beispiel geprüft, ob jedes Use-Case Diagramm genau einen auslösenden Aktor und mindestens ein verbundenes Sequenzdiagramm besitzt.

Wie schon erwähnt, erlaubt das Programm das Anlegen von mehreren gleichnamigen Klassen in einem Klassendiagramm, wobei nur von einem grafischen Element (dem zuerst angelegten) aus der Zugriff auf die Eigenschaften der Klasse möglich ist. Werden Modellierungselemente aus einem Diagramm gelöscht, werden diese nicht aus der Datenbasis gelöscht. Der Browser stellt den Anzeigemodus *Unreferenced Class Definitions* zur Verfügung, mit dem alle im Repository vorhandenen, jedoch nicht verwendeten Klassen angezeigt werden können.

e) Report-Ausgabe

Report-Ausgabe, S. 43

1.	+
2.	+
3.	+
4.	+
5.	0
6.	0
7.	+
8.	-
9.	+
10.	+

Cool:Jex beinhaltet einen Report- und einen Dokumentgenerator. Reports basieren auf Scripts. Sie holen sich Daten aus dem Sterling Repository, bearbeiten und formatieren diese und geben sie in einem Report-Fenster aus. Reports werden ausschließlich als ASCII-Text ausgegeben.

Standardmäßig steht dem Benutzer eine Vielzahl von vorgefertigten Reports zur Verfügung, die er durch einen Menübefehl starten kann. Dies sind zum Beispiel:

- *Class Hierarchy*
  Der Report Listet alle vorhandenen Klassen zusammen mit ihren Unterklassen auf.
- *Graphical Component*
  Der Report Listet alle grafischen Komponenten auf und gibt die Diagramme an, in denen sie vorkommen.
- *Objects/x*
  x kann für Klassen, Zustände, Aktivitäten, Aktoren, Instanzen und Anwendungsfälle stehen, und der Report listet Informationen zu allen gefundenen Objekten dieses Typs auf.

- *Communication*
  Der Report erlaubt die Auswahl von ein oder mehreren Beziehungstypen, und listet dann alle gefundenen Beziehungen dieses Typs (dieser Typen) auf.
- *Metric Information*
  Der Report gibt Statistiken über das Modell aus.
- u.v.m.

Die vorgefertigten Reports bieten kaum Möglichkeiten zur Anpassung. Es ist jedoch möglich, selbst Reports zu entwerfen, indem man Scripts für diese erstellt. Cool:Jex unterscheidet zwischen prozeduralen und objektorientierten Reports. Prozedurale Reports werden in *Tcl* (*Tools Command Language*) geschrieben und erlauben ausschließlich Zugriff auf Browser-Objekte (Projekte, System-Versionen, Files etc.). Objektorientierte Reports sind komplexer, bieten jedoch auch wesentlich mehr Möglichkeiten. Bei dieser Art von Reports kommt *Object-Tcl*, eine objektorientierte Erweiterung der Tcl zum Einsatz. Alle verwendeten Klassen werden von bestehenden Klassen einer Hierarchie abgeleitet und erben so deren Funktionalität. Mit objektorientierten Reports kann sowohl auf Browser-Objekte, als auch auf Diagramm-Objekte (Klassen, Objekte, Use-Cases) und deren Eigenschaften zugegriffen werden. Alle mitgelieferten Reports sind in Object-Tcl geschrieben.

Reports, die den Benutzer über die Ergebnisse einer Konsistenzprüfung informieren, existieren in dieser Form nicht. Allerdings wird bei einer Konsistenzprüfung ein Ausgabefenster angelegt und in diesem Informationen über den Arbeitsfortschritt und eventuelle Fehler ausgegeben.

Die zweite Möglichkeit an Informationen über das Modell zu gelangen, ist der Dokumentgenerator. Die so erstellten Dokumente sind wesentlich umfangreicher als Reports. Der Dokumentgenerator kann über Module an gängige Textverarbeitungssysteme wie MS-Word oder FrameMaker angebunden werden, wodurch eine aufwendige Formatierung dieser Dokumente möglich wird.

Der Inhalt eines Dokumentes ist durch eine Strukturdatei festgelegt. Der Benutzer kann die mitgelieferten Strukturdateien für Modellierungs- und Implementationsinformationen verwenden, oder die Modellbeschreibung selbst strukturieren.

Die Dokumente, die der Generator mit den mitgelieferten Schablonen für MS-Word 97 generiert sind recht gut gelungen. Es werden mehrere Dateien angelegt, die über Links miteinander verbunden sind. In den Text sind Bitmap-Dateien eingefügt, welche die Diagramme des Modells darstellen. Die Dokumente sind schlicht und übersichtlich formatiert.

f) Datenbankschema-Ausgabe

Datenbankschema-Ausgabe, S. 43

1.	+
2.	-
3.	-
4.	+
5.	-
6.	+

Cool:Jex ist in der Lage, aus den Informationen des Repositorys ein Datenbankschema zu generieren. Hierzu muß mindestens eine Klasse im Modell *persistent* erklärt werden. Wird nun wie gewöhnlich die Codegenerierung gestartet, werden zusätzlich mehrere Dateien mit SQL-Code erstellt, die Datenbanktabellen und -prozeduren erstellen. Die Erstellung eines EER-Diagramms ist in Cool:Jex nicht möglich.

Das Programm ermöglicht dem Benutzer schon bei der Modellierung für Attribute anzugeben, ob es sich bei diesen um Schlüsselattribute handelt. Eine Domainangabe für Attribute ist nicht vorgesehen.

Der von Cool:Jex generierte SQL-Code berücksichtigt alle relevanten Informationen wie Attribute und deren Datentyp, Beziehungen, usw. der statischen Klassenstruktur.

g) Diagramm-Ausgabe

Diagramm-Ausgabe, S. 44

1.	+
2.	+
3.	-
4.	+
5.	0
6.	+

Das Ansprechen verschiedener Drucker macht in der getesteten Windows-Version erwartungsgemäß keine Probleme. Cool:Jex stellt dem Benutzer zwar nicht den von Windows vorgesehenen Dialog für die Druckereigenschaften zur Verfügung, sondern verwendet ein eigenes Setup-Fenster, in dem lediglich der zu benützende Drucker und die Papiergröße eingestellt werden kann, jedoch kann jeder von Windows erkannte Drucker verwendet werden.

Die Qualität der Ausdrucke ist gut, eine Auswahl zwischen höherer Geschwindigkeit oder höherer Qualität kann im Programm nicht getroffen werden. Da Cool:Jex die von Windows zur Verfügung gestellten Drucker-Eigenschaften nicht verwendet, kann auch auf diesem Weg nicht die Geschwindigkeit oder Qualität variiert werden.

Das Programm erlaubt es, den Ausdruck kachelartig auf mehrere Blätter zu verteilen. Der Benutzer kann entweder einen Skalierungsfaktor einstellen, womit sich die Größe des Ausdruckes ergibt, oder die gewünschte Ausdruckgröße vorgeben, wobei das Diagramm vom Werkzeug automatisch skaliert wird. Die gewünschte Größe des Ausdruckes gibt der Benutzer als Anzahl der Druckseiten in horizontaler und vertikaler Richtung an. Der Diagrammausdruck erfolgt wahlweise im Hoch- oder Querformat

Das beim Drucken erhaltene Ergebnis entspricht genau der Anzeige auf dem Bildschirm, es sei denn, der Benutzer wählt den Aufdruck einer Textbox. In diesem Fall ergänzt das Programm den Ausdruck um ein rechteckiges Feld, in dem Diagrammname, Änderungsdatum, Status und Seitennummer angegeben werden.

Der Detaillevel des Ausdruckes läßt sich bei den Druckeinstellungen nicht manipulieren. Nimmt man jedoch Änderungen an der Ansicht im Grafikeditor vor, wirkt sich dies genauso auf den Ausdruck aus.

h) Offenheit - Import / Export

Offenheit - Import / Export, S. 44

1.	+
2.	+
3.	-
4.	-
5.	+
6.	-
7.	0
8.	0

Sterling Software vertreibt einige weitere Programme. Mit Cool:Jex arbeiten die folgenden beiden Programme der Firma zusammen:

- ❑ Cool:BusinessTeam, ein grafisches Daten- und Prozeßmodellierungswerkzeug
- ❑ Cool:DBA, ein Datenmanagement- und Datenbankdesignwerkzeug
- ❑ Folgende Software von Drittanbietern arbeitet mit Cool:Jex zusammen (laut Sterling Software):
- ❑ Konfigurationsmanagement: Continuous/CM, ClearCase, PVCS, Razor/CM, APEX Ada
- ❑ Dokumenterstellung: MS-Word, FrameMaker, Interleaf
- ❑ Anforderungsmanagement: DOORS, icConcepts RTM, und SLATE
- ❑ Testsoftware: QA Organizer
- ❑ Modell Im-/Export: Microsoft Repository

Das Programm ist in der getesteten Windows-Version in der Lage, mit jedem anderen Windows-Programm Daten über die Zwischenablage auszutauschen. Dies funktioniert sowohl mit Text, als auch mit grafischen Objekten aus dem Diagrammeditor, wobei letztere nach dem Einfügen in einem anderen Programm eine enorme Größe aufweisen. Es besteht weiters die Möglichkeit, Grafiken aus dem Diagrammeditor als Windows Meta File (*.wmf), Encapsulated Postscript (*.eps) oder Portable Network Graphics (*.png) zu exportieren.

Windows Standards wie OLE oder COM werden von Cool:Jex nicht unterstützt. Ebensowenig kann das Programm Reports in HTML generieren.

i) Code-Generierung

Die Codegenerierung für folgende Zielsprachen ist möglich:

- ❑ C++
- ❑ Java
- ❑ Ada
- ❑ Visual Basic
- ❑ Power Builder
- ❑ Forte
- ❑ CORBA IDL
- ❑ INFORMIX-NewEra
- ❑ Smalltalk
- ❑ DDL

Für alle im Modell enthaltenen Klassen wird im Code eine entsprechende Codierung vorgenommen. Für jede Klasse wird hierbei eine eigene Datei angelegt. Attribute werden mit Datentyp und eventuell Initialwert, Methoden mit Rückgabetyp und Parameterliste angelegt. Beziehungen werden als Zeiger auf Instanzen anderer Klassen generiert. Auch die im Modell festgelegte Sichtbarkeit wird im generierten Code berücksichtigt.

Neben der Header-Datei für die Deklaration wird für jede Klasse auch eine Code-Datei (*.cpp) angelegt, in der die leeren Hüllen der Methoden abgelegt werden, die der Benutzer händisch erweitern kann. Der Umfang und die Detailtiefe der automatischen Codegenerierung kann vom Benutzer nicht beeinflußt werden.

Wird der vom Werkzeug generierte Code händisch erweitert, bleibt bei einem neuerlichen Durchlauf des Generators der zugefügte Code erhalten. Über *Code Injection* kann auch beim Reverse-Engineering Code, der über eine Deklaration hinausgeht, von Cool:Jex erhalten werden. In diesem Fall übernimmt das Programm den Code, für den Benutzer unmerklich, in das Projekt und setzt ihn bei der nächsten Codegenerierung wieder ein.

Der Code, den Cool:Jex erzeugt, ist übersichtlich gestaltet. Alle darin enthaltenen Kommentare haben für den Benutzer eine Bedeutung, "Notizen" für das Werkzeug in Form von kryptisch anmutenden Kommentaren sind darin nicht vorhanden.

Code-Generierung, S. 45

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	+
8.	-
9.	-
10.	+
11.	-

j) Reverse-Engineering

Cool:Jex kann bestehenden Code einlesen und daraus selbsttätig Diagramme generieren. Dabei ist es theoretisch ohne Belang, ob dieser Code vom Werkzeug selbst oder auf andere Art erzeugt wurde. Beim Vorgang des Reverse-Engineerings sind für das Programm eigentlich nur die Header-Dateien mit den Deklarationen interessant. Aktiviert der Benutzer die Option *Code Injection*, wird auch Code, der über die Deklaration hinausgeht, vom Programm analysiert und ins Modell übernommen.

Reverse-Engineering, S. 45

1.	+
2.	+
3.	-

k) Test / Verifikation

Cool:Jex arbeitet mit dem Testwerkzeug QA Organizer von Segue Software zusammen. Aus einem Use-Case Diagramm werden die nötigen Informationen extrahiert und hiermit ein *.pln-File erzeugt, der die logische Teststruktur enthält, und vom QA Organizer weiterverarbeitet werden kann. Mit dem

Test / Verifikation, S. 45

1.	*0*
2.	-

Organizer werden die Tests detailliert ausformuliert, und anschließend auch überwacht. Der QA Organizer stand für diese Arbeit nicht zur Verfügung und konnte daher nicht getestet werden.

Eine Möglichkeit, die Funktionsweise des Modells animiert darzustellen, ist nicht in Cool:Jex enthalten.

l) Maßzahlen und Statistiken

Maßzahlen und Statistiken, S. 46

1.	+
2.	+
3.	-
4.	+

Der Reportgenerator von Cool:Jex ist fähig, auch Maßzahlen und Statistiken über das aktuelle Repository auszugeben. Hierzu zählen unter anderem:

- ❑ Klassendefinitionen pro Package
- ❑ Klassendiagramme pro Package
- ❑ Use-Case Diagramme pro Package
- ❑ Klassen pro Diagramm
- ❑ Use-Cases pro Diagramm
- ❑ Attribut-/Methodenanzahl von Klassen
- ❑ Anzahl der Beziehungen pro Klasse
- ❑ Vererbungstiefe von Klassenhierarchien
- ❑ Anzahl der Kinder einer Klasse

Der Browser gibt auf Wunsch Auskunft über den Projektstatus, wie Erstellungs- oder Änderungsdatum, Version, Status (aktiv, eingefroren,...), usw.

m) Multi-User - Eigenschaften

Multi-User - Eigenschaften, S. 46

1.	+
2.	+
3.	+
4.	+
5.	-
6.	-
7.	-

Cool:Jex erlaubt einem Entwicklerteam, gemeinsam an einem Repository und an einem Projekt parallel zu arbeiten. Hierbei besitzt jeder Benutzer eine eigene Konfigurations-Version. Pakete müssen mit dem Versionierungstool eingefroren werden, um für alle Mitarbeiter nutzbar zu sein. Von solchen Packages werden nun neue Versionen erstellt, die jeder Benutzer nach belieben verändern kann, ohne die Arbeit anderer zu beeinflussen. Am Ende eines Zyklus werden die verschiedenen Versionen zu einer neuen zusammengefügt und hierbei eventuell auftretende Fehler beseitigt. Ein Lokking findet bei Cool:Jex also auf Paketebene statt.

Das Programm verfügt über ein eigenes Benutzerkonzept, mit dem verschiedenen Benutzern verschiedene Rechte zugewiesen werden können. Zum Beispiel könnten Projektleiter unbeschränkten Zugriff auf das Projekt erhalten, während Designer nur Lesezugriff auf die Analyse- und Lese-/Schreibzugriff auf die Designphase erhalten. Für die Programmierer könnte letztendlich Lesezugriff auf die Design- und Lese-/Schreibzugriff auf die Implementationsphase vorgesehen werden.

Cool:Jex verfügt über keine Möglichkeiten, mit denen Benutzer mit Mitarbeitern Kommunizieren können.

n) Security

Security, S. 46

1.	0
2.	+
3.	+
4.	+
5.	+
6.	+
7.	+
8.	+
9.	+
10.	+
11.	0

Cool:Jex besitzt keinen eigenen Login-Mechanismus. Das Programm arbeitet allerdings nur auf "echten" Multi-User-Betriebssystemen, bei denen ohnehin jeder Benutzer eine Identität besitzt. Einige Aktionen beispielsweise sind nur einer Person erlaubt, die als Administrator in das System eingeloggt ist. Das Programm führt kein Änderungsprotokoll, jedoch ist eine volle Versionsverwaltung inkludiert, die bei Veränderungen den verantwortlichen Benutzer registriert.

Es besteht die Möglichkeit, Teile des Modells "einzufrieren" und so vor Veränderungen zu bewahren. Diese Aktion ist für Konfigurations-, Phasen- und Paketversionen verfügbar.

Das Programm ist in der Lage, komprimierte Backups einzelner Projekte oder des gesamten Repositorys anzulegen.

Cool:Jex kann durch Fehleingaben nicht in einen undefinierten Zustand gebracht werden. Allen Löschungen geht eine Bestätigungsabfrage voraus.

o) Multi-Projekt-Eigenschaften / Versionsmanagement

Multi-Projekt-Eigenschaften / Versionsmanagement, S. 47

1.	*+*
2.	*+*
3.	*-*
4.	*+*
5.	*0*

Das Programm ist ohne weiteres in der Lage, mehrere Projekte zeitlich parallel abzuwickeln. Obwohl theoretisch die Möglichkeit besteht, mehrere Repositorys anzulegen, empfiehlt der Hersteller, die Daten aller Projekte in einem gemeinsamen Repository zu speichern, um einen Datenaustausch zwischen verschiedenen Projekten zu erlauben.

Der Browser, der den Inhalt des Repositorys als Treeview (ähnlich dem Windows-Explorer) darstellt, organisiert die Daten hierarchisch. Es existieren folgende fünf Level:

- *Corporate level*
  Projekte und allgemeine Daten
- *Project level*
  alle Daten die ein Projekt betreffen
- *Configuration level*
  beinhaltet Versionen von Objekten eines bestimmten Projektes, an dem ein bestimmter Entwickler arbeitet
- *Phase level*
  beinhaltet die Objektversionen einer Konfigurationsversion, die sich in einer bestimmten Phase befinden
- *Package level*
  beinhaltet die eigentlichen Modellierungsinformationen wie Klassen, Diagramme,...

Das Referenzieren von Modellierungsinformationen anderer Projekte ist nicht möglich, das Kopieren solcher Elemente ist durch den Umstand, daß sich alle Projektdaten in einem gemeinsamen Repository befinden, denkbar einfach.

Cool:Jex besitzt ein eingebautes System zur Versionskontrolle. Ein Paket kann in seinem jetzigen Zustand "eingefroren" werden und an einer neueren Version weitergearbeitet werden. Dies ist vor allem dann sinnvoll, wenn das Paket in seinem jetzigen Zustand bereits funktioniert, und nun tiefgreifende Veränderungen vorgenommen werden sollen. Falls mit der überarbeiteten Version grobe Probleme auftreten, kann der Benutzer immer noch auf die alte, funktionierende Version zurückgreifen.

5.1.3 Ergonomie

a) Benutzeroberfläche

Der getesteten Windows-Version von Cool:Jex merkt man an, daß das Programm nicht ausschließlich für dieses Betriebsystem entwickelt wurde. Es existiert eine UNIX-Version des Programmes, die sicher einigen Einfluß auf

die Windows-Variante gehabt hat. Besonders das Erscheinungsbild der grafischen Oberfläche ist ein Indiz hierfür.

Das Programm sieht nicht wie eine typische Windows-Anwendung aus. Es werden zwar Steuerelemente wie Tree- und Listen-Controls verwendet, diese verhalten sich jedoch etwas anders und hinterlassen so bei einem geübten Windows-User einen ungewohnten Eindruck. Auch die Schriftart, die Cool:Jex in Controls und Dialogen verwendet, entspricht nicht dem Standard. Weiters läßt sich das Programm an einigen Stellen nicht so bedienen, wie man dies von einem Windows-Programm erwarten würde, zum Beispiel ist für manche Einträge in Listen kein Kontextmenü verfügbar (z.B. im *Stereotype Definition Editor*). Insgesamt wirkt die Oberfläche von Cool:Jex etwas veraltet. Das Look-and-Feel von Windows, zumindest das der aktuellen Versionen 95/98 bzw. NT 4.0, wird kaum vermittelt.

Benutzeroberfläche, S. 47

1.	0
2.	0
3.	0
4.	+
5.	+
6.	0
7.	+

Die Struktur der Menüleiste entspricht dem Windows-Standard (File, Edit, View, ..., Help). Die Dialoge von Cool:Jex sehen zwar etwas ungewohnt aus, was unter anderem an der verwendeten, nicht standardmäßigen Schriftart liegt, sind jedoch übersichtlich und lassen sich zweckmäßig bedienen. Die Bezeichner der Menüaktionen sind im allgemeinen logisch und selbstbeschreibend. Zur Zeit nicht nutzbare Menüeinträge sind inaktiv und werden gegraut dargestellt.

Bei längeren Aktionen wird der Mauszeiger in eine Sanduhr verwandelt, beziehungsweise wird der Benutzer über den Fortschritt solcher Aktionen über ein Log-Fenster informiert, in dem einzelne Arbeitsschritte und Fehler aufgelistet werden (z.B. bei Konsistenzprüfung).

b) Tool-Dokumentation

Die Dokumentation zu Cool:Jex besteht aus drei dünnen gedruckten Heften und einem Online-Manual im HTML-Format, das zugleich als Hilfesystem fungiert. Die gedruckten Hefte sind ein Installationshandbuch, ein *System-Administration Guide* und ein Tutorial. Weiters liegen dem Programmpaket einige Kopien mit den neuesten, noch nicht in der Dokumentation verfügbaren Informationen bei.

Tool-Dokumentation, S. 48

1.	+
2.	+
3.	-
4.	-
5.	+
6.	0
7.	-
8.	+
9.	+
10.	+
11.	-
12.	+

Das Tutorial führt in viele Teilbereiche des Programms wie Modellierung, Versionsmanagement, parallele Entwicklung und Codegenerierung ein. Es ist einfach und verständlich verfaßt und trägt sehr gut zu einem schnellen Verständnis der Grundkonzepte des Programmes bei. Mit Cool:Jex werden keine Demobeispiele geliefert.

Die Online-Dokumentation im HTML-Format ist ebenfalls recht gut gelungen, jedoch bei der getesteten Version von Cool:Jex leider etwas fehlerhaft (siehe *Hilfefunktionen*, S. 65). Das Online-Handbuch kann aus dem Tool heraus aufgerufen werden.

c) Usability

Usability, S. 48

1.	0
2.	0
3.	+
4.	-
5.	+
6.	+
7.	+
8.	+
9.	+
10.	+
11.	-

Im Fehlerfall macht Cool:Jex den Benutzer zwar auf diesen aufmerksam, bietet jedoch keine weitere Hilfestellung an. Längere Aktionen, wie die Codegenerierung, können nicht unterbrochen werden.

Das Programm stellt eine große Anzahl von Tastatur-Shortcuts, besonders für Editier- und Dateibefehle, bereit. Eine Belegung von Funktionstasten kann nicht durchgeführt werden.

Das Programm verfügt über mehrere Werkzeugleisten und über Kontextmenüs in fast jeder Situation. Menüs und Werkzeugleisten können vom Benutzer auf relativ einfachem Weg an dessen individuelle Bedürfnisse angepaßt werden. Am unteren Fensterrand von Browser und Diagrammeditor ist

eine Statuszeile angebracht, die hilfreiche Auskünfte zu Menübefehlen oder Programmsituationen gibt.

Cool:Jex besitzt kein Hauptrahmenfenster. Sowohl der Browser, als auch die Diagrammeditoren sind kleine skriptbasierte Programme, die eine Sicht auf die Daten des Repositorys bieten und in einem eigenen Fenster ablaufen.

Das Programm ist ausschließlich in einer englischen Version erhältlich.

d) Hilfefunktionen

Hilfefunktionen, S. 49

1.	+
2.	+
3.	-
4.	+

Cool:Jex beinhaltet ein kontextsensitives Hilfesystem, welches auf vielen einzelnen Dokumenten im HTML-Format aufbaut. Alle wichtigen Dialoge besitzen einen Hilfe-Button, mit dem der Benutzer an eine bestimmte Stelle in der Online-Dokumentation gebracht wird. Die Online-Hilfe ist außerdem über einen Menüeintrag zu erreichen, ein Tastatur-Shortcut hierfür ist jedoch nicht vorhanden.

Bei der getesteten Version funktionierte dieses Hilfesystem leider nicht, da die Links innerhalb des HTML-Dokuments, sowie die kontextbezogenen Einsprünge vom Programm aus nicht stimmten. Da das Hilfesystem bis dato auf einem Dynatext-Dokument basierte und die HTML-Version erst vor kurzem, bei der Version 4.1, eingeführt wurde, kann wohl davon ausgegangen werden, daß es sich hierbei um einen Flüchtigkeitsfehler handelt, und dieser bereits durch Patches oder ähnliches behoben wurde.

Ein anderes Problem ist, daß das HTML-Hilfesystem zwar angenehm zu lesen ist und viele informative Grafiken enthält, jedoch keine Suche nach einem bestimmten Stichwort zuläßt.

Zusätzlich verfügt das Programm über Hot-Help. Wird der Mauszeiger über einen Button einer beliebigen Werkzeugleiste bewegt, wird ein Bezeichner angezeigt, der das Werkzeug, das sich hinter dem Symbol verbirgt, beschreibt.

e) Graphikeditor / Darstellung

Graphikeditor / Darstellung, S. 49

1.	-
2.	-
3.	+
4.	+
5.	-
6.	+
7.	-
8.	-
9.	+
10.	*0*
11.	-
12.	+
13.	+
14.	-
15.	+

Der Grafikeditor von Cool:Jex ist ein sehr mächtiges Werkzeug. Auch speziellere Konstrukte der UML, wie zum Beispiel Design Patterns, nebenläufige Zustände oder synchrone und asynchrone Nachrichten können modelliert werden. Ein Anpassen der vorhandenen Symbole oder das Zeichnen von freien Formen, um nicht unterstützte Notationselemente nachzubilden, ist nicht möglich. Die Darstellung auf dem Bildschirm entspricht dem späteren Ausdruck.

Der Editor gestattet dem Benutzer, für alle Symbole eine Linien-, Füll- und eine Textfarbe festzulegen. Ebenfalls einstellbar, allerdings nur auf Diagrammebene und nicht für jedes Element einzeln, ist die Schriftart für die Symbolbeschriftung. Symbole und Text können nicht rotiert werden.

Das Ansichtsfenster des Diagrammeditors kann in feinen Stufen gezoomt werden. Eine Funktion, um ein vorhandenes Diagramm in das aktuelle Fenster einzupassen und dementsprechend Zoomstufe und Position zu verändern, ist nicht verfügbar.

Um das Diagrammlayout übersichtlicher zu gestalten, erlaubt der Editor dem Benutzer in bestehende Beziehungen nachträglich Eckpunkte einzufügen. Dies geschieht über das Werkzeug *Vertex*, das zuvor aus der Werkzeugleiste ausgewählt werden muß. Cool:Jex bietet keine Funktionen, die den Anwender bei der Layoutoptimierung unterstützen. Weder eine automatische Layoutroutine für das gesamte Diagramm, noch entsprechende Alignment-Funktionen zur Ausrichtung gewählter Elemente sind verfügbar. Die

Notationselemente erhalten bei ihrem Anlegen im Diagramm eine vorgegebene Größe, diese wird jedoch automatisch angepaßt, wenn zum Beispiel lange Bezeichner, oder mehrere Attribute oder Methoden eingegeben werden. Ein Zeilenumbruch innerhalb der Bezeichnung eines Elementes ist möglich. Das Layout eines Diagramms wird bei Verlassen des Grafikeditors gespeichert. Diese Speicherung wird nicht im Repository, sondern in einzelnen Diagrammdateien vorgenommen.

f) Graphikeditor / Bedienung

Graphikeditor / Bedienung, S. 50

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	+
8.	+
9.	+
10.	+
11.	+
12.	+
13.	+
14.	+

Für das Anlegen von Notationselementen stehen Werkzeuge bereit, die über eine seitlich am Fenster angebrachte Werkzeugleiste ausgewählt werden. Ist ein Werkzeug selektiert, kann es mit einem einfachen Mausklick in das Diagramm eingezeichnet werden. Der Cursor für die Tastatureingabe wird nach dem Neuanlegen eines Elementes in das Diagramm gesetzt und es kann sofort der Bezeichner eingegeben werden. Wird dieser nicht festgelegt, bleibt das Element namenlos, das heißt, es wird kein Default-Name vergeben. Das gewählte Werkzeug bleibt bis zur Auswahl eines anderen aktiv. Es deaktiviert sich nach einer vollzogenen Zeichenaktion nicht selbständig.

Einmal angelegte Notationselemente können per Drag&Drop im Diagrammfenster verschoben werden. Diese Verschiebung kann frei, oder auf einem Punktraster erfolgen. Die Körnigkeit dieses Rasters kann vom Benutzer verstellt werden. Auch Beziehungen können im Nachhinein manipuliert werden. Es können sowohl neue Eckpunkte eingefügt werden, um zum Beispiel Überschneidungen zu vermeiden, als auch die an der Beziehung beteiligten Notationselemente verändert werden, indem ein Ende einer Relation per Drag&Drop mit der Maus von dem zugehörigen Symbol weggezogen und mit einem anderen verbunden wird. Der Grafikeditor schränkt den Benutzer beim Positionieren der Symbole nicht ein. Alle Elemente einschließlich Beziehungen können genau plaziert werden.

Die Zeichenfläche ist unendlich groß und wird über seitliche Scrollbalken positioniert.

Das Anwählen mehrerer Objekte ist durch Ziehen eines Rahmens mit der Maus, bzw. durch Anklicken bei gehaltener Strg-Taste möglich. Auf diese gemeinsam gewählten Elemente können zugleich Editierfunktionen wie Verschieben oder Löschen ausgeführt werden.

Ein Abbruch von mehrteiligen Zeichenoperationen, wie das Anlegen von Beziehungen, ist jederzeit mit der rechten Maustaste möglich.

Der Editor enthält eine Undo- jedoch keine Redo-Funktion. Außerdem stehen die üblichen Zwischenablagefunktionen (Cut, Copy, Paste,...) zur Verfügung, die auch mit den gebräuchlichen Shortcuts erreichbar sind.

Um die Arbeit im Diagrammeditor zu beschleunigen existiert ein sogenannter *Pointer Focus*-Modus, der bei Bedarf aktiviert werden kann. Hierbei wird der Eingabefokus sofort auf das jeweilige Textelement gesetzt, das sich unter dem Mauszeiger befindet. Ein Mausklick, um dieses zu markieren, entfällt somit.

g) Performance

Performance, S. 51

1.	+
2.	+
3.	+
4.	+

Die Performance von Cool:Jex ist in Anbetracht seines erheblichen Funktionsumfanges außerordentlich zufriedenstellend.

h) Adaptivität

Die Oberfläche des Programms läßt sich individuell anpassen. Auf einfache Weise können Menüstrukturen und Werkzeugleisten verändert werden, jedoch kann theoretisch mit der äußerst mächtigen Skriptsprache *TCL* (*Tools Command Language*) ein völlig neuer Browser oder Graphikeditor erstellt werden.

Cool:Jex unterstützt keine Makros und wird nicht mit dem Quellcode ausgeliefert.

Adaptivität, S. 51

1.	-
2.	+
3.	+
4.	-
5.	+
6.	-

5.2 objectiF 3.2b

Das Programm objectiF der deutschen Firma MicroTOOL GmbH ist im Vergleich zu den meisten seiner Mitbewerber ein eher einfaches Werkzeug. Das installierte Programm belegt nur knappe 18 Mb auf der Festplatte (inklusive Dokumentation, exklusive Demoprojekte). In Bezug auf Funktionsumfang und gestalterischen Möglichkeiten kann es nicht mit den großen CASE-Tools mithalten. Dennoch ist objectiF ein leicht zu bedienendes Programm, das bei kleineren Projekten, oder solchen, die mit seinem eingeschränkten Funktionsumfang auskommen, gute Dienste leisten kann. MicroTOOL macht keine Angaben darüber, welcher Version des UML-Standards das Programm nachempfunden wurde.

5.2.1 Methodenbasierte Kriterien

a) Modellierung der Objektstruktur - Objektstatik

objectiF verwaltet einen Datenstamm, in dem alle Informationen gespeichert sind. Werden Eigenschaften eines vorhandenen Objektes direkt in der Datenstamm-Ansicht, oder in einem beliebigen Diagramm geändert, wirkt sich dies automatisch auf alle Diagramme aus, die dieses Objekt enthalten.

Es wird beispielsweise im Code-Fenster, in dem die Klassendeklaration einer bestimmten Klasse dargestellt wird automatisch eine neue Variable deklariert, wenn diese in beliebiger Weise in das Modell eingeführt wird.

Bestehende Objekte eines Modells können an beliebiger Stelle des Programms über einen Eigenschaften-Dialog bearbeitet werden. Der Datenstamm des Modells wird immer konsistent gehalten. Das Werkzeug verhindert zum Beispiel die nochmalige Vergabe desselben Klassennamens durch Deaktivierung des *OK*-Buttons des Eigenschaftendialoges. Der Benutzer muß so entweder den Namen abändern, oder den Dialog über den Abbrechen-Button verlassen.

Die Klassendiagramm-Ansicht stellt die Klassen als ausgefüllte Rechtecke dar, die nur den Klassennamen enthalten. Optional kann der Anwender den unteren Rand einer Klasse mit der Maus herunterziehen und so die Attribute und Methoden der Klasse sichtbar machen. Die Reihenfolge der Komponenten einer Klasse spiegelt ihre Reihenfolge bei der Erstellung wider und kann darüberhinaus nicht beeinflußt werden.

Für angelegte Attribute muß ein Datentyp angegeben werden, hierbei ist es möglich benutzerdefinierte Typen zu definieren. Für Methoden können

Modellierung der Objektstruktur - Objektstatik, S. 38

1.	+
2.	+
3.	-
4.	+
5.	+
6.	+
7.	-

Rückgabetypen angegeben werden, diese können jedoch weggelassen werden. Für Methoden können weiters Parameter angegeben werden, die wie Attribute einen Datentyp erhalten müssen. Zusätzlich zu den Datentypen der Parameter können diese als Zeiger, Referenz oder Konstante definiert werden. Die Eingabe der Parameterliste erfolgt textuell. objectiF überwacht die Eingabe und blockiert bei syntaktischen Fehlern oder unsinniger Eingabe den *OK*-Button des Dialoges. Vor- und Nachbedingungen für Methoden können nicht angegeben werden.

b) Modellierung der Systemstruktur - Systemstatik

Modellierung der Systemstruktur - Systemstatik, S. 39

1.	+
2.	0
3.	-
4.	-

Es werden die Beziehungstypen Assoziation, Aggregation und Generalisation zur Verfügung gestellt. Assoziation und Aggregation können auch rekursiv definiert werden.

objectiF erlaubt die nachträgliche Manipulation von Beziehungen hinsichtlich ihrer Multiplizität, Rollen und Navigierbarkeit, es ist jedoch unmöglich, einen Beziehungstyp nachträglich in einen anderen umzuwandeln. Ebenso unmöglich ist es, eine der beiden Klassen, die miteinander in Beziehung stehen, nachträglich durch eine andere zu ersetzen.

Das Programm erlaubt keine Strukturierung der Komponenten außer Packages, die auch geschachtelt eingesetzt werden können.

objectiF bietet keine OCL-Unterstützung.

c) Dynamisches Objektverhalten - Objektdynamik

Dynamisches Objektverhalten - Objektdynamik, S. 39

1.	+
2.	-
3.	+
4.	-
5.	-
6.	-

Jeder Klasse kann ein Zustandsdiagramm zugeordnet werden, wobei der Bezug zur Klasse jederzeit eindeutig ist. Das Tool ordnet jedem Übergang von einem in einen anderen Zustand eine Methode des betreffenden Objekts zu. Es können zwar Bedingungen für Zustandsübergänge modelliert werden, es kann hierbei jedoch nicht explizit auf Attribute des Objektes Bezug genommen werden. Es ist nicht möglich, Aktivitäten und Ereignisse im Sinne einer genaueren Beschreibung zu verfeinern.

Generell sind die Möglichkeiten, die das Programm zum Anlegen von Zustandsdiagrammen bietet, eher bescheiden. Es sind keine geschachtelten Zustände und auch keine Generalisierung von Ereignissen möglich.

d) Dynamisches Systemverhalten - Systemdynamik

Dynamisches Systemverhalten - Systemdynamik, S. 39

1.	-
2.	0

Die Aggregation von Zuständen nebenläufiger Objekte kann nicht modelliert werden.

Im Sequenzdiagramm kann auf die Methoden, nicht jedoch auf die Attribute des statischen Objektmodells Bezug genommen werden.

e) Zusammenhang zwischen den Modellen

Zusammenhang zwischen den Modellen, S. 39

1.	+
2.	-
3.	+
4.	-

Jedes Zustandsdiagramm ist einer bestimmten Klassen zugeordnet, das dynamische Verhalten von Superklassen wird jedoch nicht an deren Subklassen weitervererbt.

Bei der Modellierung im dynamischen Modell kann auf die Methoden, nicht jedoch Attribute des statischen Modells Bezug genommen werden.

f) Abstraktionsebenen / Sichten

Abstraktionsebenen / Sichten, S. 40

1.	0
2.	-
3.	-
4.	-
5.	-

objectiF stellt im Klassendiagramm per Standardeinstellung Klassen nur als Rechtecke mit dem Klassennamen dar. Informationen über die zur Klasse gehörigen Attribute und Methoden erhält man, wenn man die dargestellte Klasse mit der Maus nach unten hin "aufmacht".

Im Klassendiagramm und bei wie oben beschrieben geöffneten Klassen werden Attribute mit ihrem Datentyp und Methoden mit Rückgabetyp und Parameterliste angezeigt. Es ist nicht möglich, den Detaillierungsgrad der Diagrammansicht so zu verändern daß, zum Beispiel keine Datentypen mehr oder nur öffentliche Komponenten angezeigt werden.

Das Programm bietet keine Möglichkeit, die Detailtiefe bei Sequenz- oder Zustandsdiagramm einzustellen.

g) Vorgangsreihenfolge bei der Entwicklung

Vorgangsreihenfolge bei der Entwicklung, S. 40

1.	+
2.	+
3.	+
4.	-

Bei der Modellierung gestattet das Werkzeug dem Anwender, neue Modellierungselemente zunächst nur grob zu spezifizieren (Namen) und die Angaben erst später genauer auszuführen. Bei der Schaffung eines neuen Objektes gelangt man in dessen Eigenschaften-Dialog, in dem nur der Name eingegeben werden muß. Alle anderen Parameter sind optional oder werden auf Standardwerte gesetzt.

objectiF erlaubt es auch direkt in der Repository-Ansicht, die als Treeview ausgeführt ist, Modellelemente einzufügen oder zu löschen. So eingefügte Objekte können zu einem späteren Zeitpunkt in ein beliebiges Diagramm übernommen werden.

Objekte wiederum, die in Diagrammen neu angelegt werden, werden sofort in den Datenstamm übernommen. Beim Löschen von Modellelementen aus einem Diagramm muß der Anwender entscheiden, ob nur die Referenz aus dem Diagramm, oder das Objekt selbst gelöscht werden soll, wobei im letzteren Fall die Löschung im Repository vorgenommen wird.

h) Unterstützung bei der Methodenanwendung

Unterstützung bei der Methodenanwendung, S. 40

1.	0
2.	-
3.	-
4.	0
5.	-
6.	-

Zu objectiF wird keine explizit die UML erklärende Dokumentation mitgeliefert, jedoch wird in der Online-Hilfe und den Tutorials am Rande darauf eingegangen. Die meisten Notationselemente in objectiF entsprechen genau der Vorgabe durch die UML-Notation, wodurch die Lektüre von Standardwerken zu diesem Thema Sinn macht.

5.2.2 Funktionale Kriterien

a) Datenhaltung

Datenhaltung, S. 41

1.	+
2.	+
3.	+
4.	+
5.	-
6.	+
7.	-
8.	-

objectiF speichert die Informationen zu einem Modell in einem Data-Dictionary, durch das mit einem Browser, der Windows-typisch als Treeview ausgeführt ist, navigiert werden kann. Informationen werden in der Datenbasis nicht redundant gespeichert. Kommt zum Beispiel eine Klasse in mehreren Klassendiagrammen vor, wird diese Klasse zwar im Browser zweimal angezeigt (als Bestandteil jedes der beiden Klassendiagramme), jedoch verweisen die beiden Einträge auf ein einziges Objekt.

Es ist möglich, Objekte mittels Browser direkt im Datenstamm anzulegen. Diese können später in Diagramme eingefügt werden. Ebenso werden

beim Reverse-Engineering die Modellinformationen zunächst nur ins Repository aufgenommen. Selbstverständlich ist es auch möglich, Objekte direkt aus dem Repository zu entfernen.

Das System gestattet den Multi-User-Betrieb über ein Netzwerk. Für sicheren Betrieb und die Konsistenz solcher Projekte sorgt ein integriertes OODBMS. objectiF erlaubt es nicht Projektdaten in Dateiformaten zu speichern, die von gängigen Datenbanksystemen gelesen werden können. Es existieren auch keine Schnittstellen zu relationalen Datenbanksystemen. Es ist ferner unmöglich, Daten in ASCII-Text zu exportieren.

b) Browsing-Eigenschaften

Browsing-Eigenschaften, S. 41

1.	-
2.	-
3.	-
4.	-
5.	-
6.	-
7.	+
8.	+
9.	-
10.	+
11.	-

Die einzige Suchfunktion, die objectiF dem Anwender bietet, ist die Möglichkeit, im Browser über die Tastatur einen oder mehrere Buchstaben einzugeben. In diesem Fall wird das nächste Element im Browserfenster markiert, dessen Name mit dem eingegebenen Teilstring übereinstimmt. Diese Funktion arbeitet exakt wie die des Windows-Explorers. Weitere Suchfunktionen sind nicht verfügbar. Werden die Eigenschaften von Objekten im Browser verändert, werden sofort eventuell vorhandene geöffnete Diagrammansichten aktualisiert.

objectiF stellt dem Benutzer sowohl im Browser, als auch in der Klassendiagrammansicht, die Funktion *Klassenhierarchie anzeigen* zur Verfügung. Diese Funktion zeigt in einem separaten Browserfenster alle Superklassen einer gewählten Klasse an.

Mit der Funktion *Referenzen* erhält der Benutzer darüber Auskunft, in welchem Zusammenhang ein ausgewähltes Element verwendet wird. Es ist möglich Referenzen über Attribute, Methoden oder Klassen anzuzeigen.

c) Navigation im Systemmodell

Navigation im Systemmodell, S. 42

1.	+
2.	+
3.	-
4.	+

Das Hauptfenster von objectiF besteht nur aus einer Menü- und einer Iconleiste. Kindfenster, die für Diagramm- und Browseransichten anlegt werden, plaziert das Programm frei auf dem Desktop. Die Navigation zwischen den Ansichten geschieht entweder händisch, durch ein Anwählen der betreffenden Ansichten mit Maus oder Task-Manager, oder durch ein Browser Fenster von objectiF.

Bestimmte Diagramme können oder müssen anderen zugeordnet werden. Zustandsdiagramme können zum Beispiel ausschließlich zu einer bestehenden Klasse angelegt werden. Sequenzdiagramme können ungebunden erstellt werden, es ist jedoch möglich, diese bestimmten Klassendiagrammen zuzuweisen. objectiF erleichtert die Navigation zwischen solcherart abhängigen Diagrammen, indem es den Wechsel von einem Diagramm oder Objekt zu dessen zugeordneten Diagrammen via Kontextmenü ermöglicht. Abseits der vom Programm vorgegebenen Abhängigkeiten zwischen Diagrammen (Klasse - Zustandsdiagramm, Sequenzdiagramm - Klassendiagramm,...) ist es nicht möglich, eigene Abhängigkeiten festzulegen.

d) Konsistenzprüfung / Integrität

Eine Funktion, die ein bestehendes Modell auf Konsistenz prüft, ist nicht in objectiF enthalten. Um die Konsistenz der Projekte dennoch sicherzustellen, versucht das Programm ungültige Eingaben gar nicht zuzulassen. Wird zum Beispiel eine neue Klasse angelegt, gelangt man in den Eigenschaftendialog dieser Klasse, der erst verlassen werden kann, wenn zumindest ein seman-

tischer und ein technischer Name angegeben wurde. Um die Eingabe zu beschleunigen, übernimmt das Programm den eingegebenen semantischen Namen leicht modifiziert als Voreinstellung für den technischen Namen (z.B. Klasse *Linie*, technischer Name *CLinie*). Ebenso wird überwacht, ob der Name eines Elementes schon im Modell existiert, und gegebenenfalls der *OK*-Button des Dialoges deaktiviert, um den Benutzer an der Bestätigung für die falsche Eingabe zu hindern. Zu beachten ist, daß nur die technischen Namen auf Gleichheit geprüft werden. Das Modell darf also mehrere Elemente mit gleichem semantischen Namen enthalten, solange deren technische Namen unterschiedlich sind. Es ist nicht möglich, namenlose Modellierungselemente zu erstellen.

Konsistenzprüfung / Integrität, S. 42

1.	+
2.	+
3.	+
4.	+
5.	+
6.	-
7.	-
8.	-
9.	+

Auch bei der Erstellung von Beziehungen wacht das Programm darüber, daß die Konsistenz des Modells erhalten bleibt. Es ist beispielsweise nicht möglich, Relationen ins Leere gehen zu lassen, oder Generalisationen auf sich selbst darzustellen.

Will der Anwender ein Element aus einem Diagramm löschen, wird er vor die Wahl gestellt, das Objekt selbst, oder nur das Symbol aus dem Diagramm zu löschen. Im ersteren Fall wird das Objekt aus der Datenbasis, und damit auch aus allen Diagrammen in denen es vorkommt, gelöscht. Im zweiten Fall wird lediglich die Darstellung des Objektes im aktuellen Diagramm gelöscht, das Repository und andere Diagramme bleiben unberührt. Hierbei ergibt sich die Möglichkeit, daß Elemente, die in keinem Diagramm vorkommen unbemerkt im Repository verbleiben und dort Platz verbrauchen. Leider bietet objectiF keine Möglichkeit, solche Elemente zu suchen und anzuzeigen. Eine etwas umständliche Methode wäre, im Repository-Browser alle Objekte der Reihe nach auszuwählen und sich die Referenzen auf diese anzusehen.

e) Report-Ausgabe

Report-Ausgabe, S. 43

1.	+
2.	+
3.	+
4.	*0*
5.	*0*
6.	-
7.	+
8.	-
9.	-
10.	+

Um Projekte, die mit objectiF erstellt werden automatisch zu dokumentieren, bietet das Programm zwei Möglichkeiten:

Bei der ersten Möglichkeit, die sich für konventionelle Reports in Papierform eignet, arbeitet objectiF via COM mit dem Textverarbeitungssystem MS-Word zusammen. Die automatische Generation solcher Reports wird durch den Menüeintrag *Auswertungen* initiiert. Hier kann der Anwender entscheiden, für welche Modellelemente bzw. -aspekte objectiF Reports generieren soll. Weiters kann festgelegt werden, wie genau der Report ausfallen soll. Beispielsweise kann der Anwender bei einer Klasse entscheiden, ob Attribute, Methoden, Beziehungen, Bemerkungen und ein möglicherweise vorhandenes zugehöriges Zustandsdiagramm in den Report aufgenommen werden soll.

Die zweite Möglichkeit gestattet dem Anwender, Reports im HTML-Format zu generieren. Diese Reports sind übersichtlich strukturiert und sind vorgesehen, um Informationen über ein lokales Netzwerk zur Verfügung zu stellen.

In beiden Arten von Reports lassen sich Aspekte der statischen Struktur (Klassen, Attribute, Methoden, Vererbungs-, Aggregations-, Assoziationsbeziehungen), sowie des dynamischen Systemverhaltens (Sequenzdiagramme) beschreiben. Eine Beschreibung des dynamischen Objektverhaltens steht nur für Reports mit MS-Word zur Verfügung. Da objectiF keine Funktion zur expliziten Konsistenzprüfung bereitstellt, können auch keine Informationen darüber in die Reports aufgenommen werden.

f) Datenbankschema-Ausgabe

Datenbankschema-Ausgabe, S. 43

1.	+
2.	-
3.	-
4.	-
5.	-
6.	*0*

Die Daten aus einem Modell in ein relationales Datenbankschema zu übertragen, geschieht in objectiF über die scriptgesteuerte Codegenerierung.

Zu dem Werkzeug wird ein SQL-Script Server mitgeliefert, der zu jeder persistenten Klasse via DDL-Code eine Tabelle generiert. Genau wie bei der Codegenerierung ist in einem Editorfenster das vom Script Server erzeugte Ergebnis sichtbar und kann dort vom Benutzer händisch editiert werden.

g) Diagramm-Ausgabe

Diagramm-Ausgabe, S. 44

1.	+
2.	+
3.	*0*
4.	+
5.	+
6.	*0*

objectiF baut als Windows-Anwendung auf der betriebssystemeigenen Druckerschnittstelle auf und kann somit jeden Drucker verwenden, der von Windows erkannt wird. Der Druckertreiber und das Betriebssystem stellen die Funktionalität bereit, mit welcher der Benutzer die Qualität des Ausdrucks und die Orientierung am Blatt bestimmen kann.

objectiF selbst bietet ebenfalls eine Reihe von Einstellmöglichkeiten, mit denen sich Qualität und Umfang des Ausdrucks verändern lassen. Der Benutzer kann die Detailtiefe variieren, indem er das Programm anweist, keine Methoden und Attribute zu drucken, oder die Beziehungskanten nicht zu beschriften. Zur optischen Aufwertung kann um das Diagramm ein Rahmen gezogen und der Ausdruck mit Autor und Titel versehen werden.

Sollen große Diagramme ausgedruckt werden, die auf einer Druckerseite keinen Platz finden, generiert objectiF auf Wunsch Klebe- und Schnittkanten, und versieht die Teilblätter mit Seitennummern.

Das Werkzeug besitzt eine Druckvorschau, die genau dem Ergebnis des Ausdrucks entspricht. Diese Vorschau kann auf Wunsch auch mehrseitige Diagrammausdrucke darstellen. Hier hat der Benutzer weiters die Möglichkeit, den Ausdruck zu zentrieren, oder auf die volle Seitengröße zu skalieren.

Die Ausdrucke, die objectiF produziert, können als präsentationsfähig bezeichnet werden.

h) Offenheit - Import / Export

Offenheit - Import / Export, S. 44

1.	+
2.	+
3.	+
4.	+
5.	+
6.	-
7.	+
8.	-

objectiF arbeitet mit der Prozeßmanagementsoftware In-Step zusammen, die wie objectiF von der Firma Microtool stammt. Mit In-Step lassen sich aus einem bestehenden Modell bestimmte Aktivitäten ableiten, und es ist möglich, diesen Termine und Ressourcen zuzuordnen. In-Step wurde im Zuge dieser Arbeit nicht getestet.

Das Programm bietet, bis auf das als Iconleiste ausgeführte Hauptfenster, Windows 95 *look and feel.* Die Browseransichten sind explorer-ähnlich als Treeview gestaltet und es kann an den meisten Stellen ein Kontextmenü aufgerufen werden. objectiF arbeitet via COM mit MS-Word zusammen, und ist weiters in der Lage, Reports in HTML zu erstellen.

Es ist nicht vorgesehen, Grafiken oder Textdokumente in ein Projekt zu importieren. Erweiterungen zu dem Programm von Drittanbietern sind den Autoren nicht bekannt, es ist jedoch möglich, diese mit einer Software-Entwicklungsumgebung zu erstellen, die COM-Komponenten erzeugen kann.

i) Code-Generierung

objectiF stellt Funktionen zur automatischen Codegenerierung bereit. Standardmäßig werden die Programmiersprachen C++ und Java unterstützt, jedoch kann das Ergebnis der Codegenerierung mit Hilfe sogenannter Code

Scripts definiert werden. So kann der Codegenerator auf jede gewünschte Programmiersprache angepaßt werden.

Bei der Codegenerierung werden zuerst auf Paketebene oder global Dateien definiert, die der Generator dann mit Code auffüllt. Hierbei kann der Anwender bestimmte Klassen des Modells den gewünschten Ausgabedateien zuordnen, um das Projekt nach seinen Wünschen zu strukturieren. Die Codegenerierung kann nach dieser Zuordnung auf Paketebene (alle Dateien, des gewählten Paketes werden generiert) oder auf Dateiebene ausgelöst werden.

Attribute werden mit Datentyp umgesetzt, Methoden mit Rückgabetyp und Übergabeparametern. Auch die Sichtbarkeit von Objekten wird im Code umgesetzt. Die Funktionsrümpfe werden von objectiF automatisch erstellt, der Programmierer muß lediglich den Code einfügen. Beziehungen zwischen Klassen werden ebenfalls automatisch codiert.

objectiF stellt einen Codeeditor zur Verfügung, der zu jedem Modellierungselement über ein Kontextmenü aufgerufen werden kann. In diesem kann der Benutzer die Methoden der im System vorhandenen Klassen manuell mit Code versehen. Der Codeeditor ist ein kleines Textverarbeitungsprogramm, das in einem eigenen Fenster ausgeführt wird. Es besitzt die übliche Funktionalität wie Cut, Copy und Paste und verfügt über eine Werkzeugleiste und eine Druckfunktion. Der Editor verfügt auch über ein Undo/Redo-System, das sogar über mehrere Ebenen arbeitet. Dies ist insofern bemerkenswert, da beim Graphikeditor, bei dem Undo-Funktionalität sicher sinnvoller gewesen wäre, auf eine solche verzichtet wurde.

Code-Generierung, S. 45

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	+
8.	-
9.	*0*
10.	+
11.	+

j) Reverse-Engineering

Es ist möglich, mit dem Werkzeug bestehenden Code einzulesen und dann weiterzubearbeiten. Dies bezieht sich auf einzelne Klassen, Klassenbibliotheken und Komponenten und der einzulesende Code muß nicht mit dem Tool selbst generiert worden sein.

Code, der mit einem anderen Editor geschrieben oder verändert wird, muß in einem eigens hierfür gekennzeichneten Block stehen, ansonsten wird er beim Reverse Engineering ignoriert. Ein solcher Block beginnt mit dem Kommentar `//...begin "Body"` und endet mit `//..end "Body"`.

Reverse-Engineering, S. 45

1.	+
2.	+
3.	+

k) Test / Verifikation

objectiF beinhaltet keine Funktionalität, die ein Testen/Verifizieren von Projekten auf Modellebene erlaubt.

Test / Verifikation, S. 45

1.	-
2.	-

l) Maßzahlen und Statistiken

Das Werkzeug bietet keine Möglichkeit, Statistiken über das Projekt auszugeben.

Maßzahlen und Statistiken, S. 46

1.	-
2.	-
3.	-
4.	-

m) Multi-User - Eigenschaften

objectiF unterstützt den Betrieb mit mehreren Benutzern gleichzeitig. Versionen für andere Betriebssysteme stehen nicht zur Verfügung.

Arbeiten mehrere Benutzer an einem Projekt, wird mittels Locking sichergestellt, daß die Konsistenz der Datenbasis erhalten bleibt. Das Lockniveau ist dabei immer das einzelne Element (z.B. Methode, Attribut...). Wenn es im Multi-User-Betrieb keine Überschneidungen gibt, dann sind auch mehrere parallele Schreibtransaktionen möglich.

Multi-User - Eigenschaften, S. 46

1.	+
2.	-
3.	-
4.	+
5.	-
6.	-
7.	-

objectiF verfügt nicht über ein eigenes Benutzerkonzept mit dessen Hilfe Benutzer in einzelne Gruppen eingeteilt werden könnten. Ebenfalls unmöglich ist die Kommunikation mit anderen Projektmitarbeitern, die gerade mit objectiF arbeiten, direkt aus dem Tool heraus.

n) Security

Security, S. 46

1.	-
2.	0
3.	-
4.	-
5.	-
6.	+
7.	+
8.	-
9.	+
10.	+
11.	-

objectiF verfügt über kein Login-Konzept und keine Rechteverwaltung. Somit ist es auch nicht möglich, Aktionen auf einen bestimmten Benutzer zurückzuführen. Benutzer werden von objectiF beim Anschluß an ein Konfigurationsmanagement-System ausgewertet. Dort können Packages für bestimmte Benutzer ausgecheckt und somit bearbeitbar gemacht werden.

Wenn eine Transaktion von objectiF nicht erfolgreich abgeschlossen werden kann, dann wird diese mit einem Rollback rückgängig gemacht. Wird die Abarbeitung einer Transaktion zum Beispiel durch einen Stromausfall unterbrochen, so wird beim nächsten Starten automatisch ein Recovery-Mechanismus aktiviert, der die Konsistenz der Datenbank wieder herstellt.

objectiF verfügt über einen Backup-Mechanismus, der auf Wunsch eine Sicherungskopie des Systems in ein Unterverzeichnis mit dem Bezeichner *Backup* speichert. Für die Sicherung kann ein Name angegeben und die Kompressionsrate über einen Schieberegler eingestellt werden. Ein System, das unkomprimiert ca. 900 kb belegte, konnte bei voller Kompression auf ein Datenvolumen von 130 kb gebracht werden. Anlegen der Sicherung bzw. Wiederherstellen eines Systems aus einer Sicherungsdatei erfolgt bequem aus dem Datei-Menü von objectiF.

o) Multi-Projekt-Eigenschaften / Versionsmanagement

Multi-Projekt-Eigenschaften / Versionsmanagement, S. 47

1.	+
2.	+
3.	-
4.	0
5.	-

Mit objectiF kann der Benutzer mehrere Projekte anlegen, und dann das jeweils gewünschte laden und bearbeiten. Ein Projekt in objectiF wird immer in einem neu angelegten Verzeichnis abgelegt und enthält weitere Unterverzeichnisse. objectiF bietet keine Möglichkeit, bestehende Projekte zu kopieren, beispielsweise als Ausgangsbasis für neue Projekte, es ist jedoch möglich, mittels eines Dateimanagers Projekte in ein Verzeichnis mit anderem Namen zu kopieren. objectiF deutet den Namen eines Projektverzeichnisses als dessen Projektnamen. Die Bezugnahme auf ein anderes Projekt ist nicht möglich, bestehende Projekte, die weiterverwendet werden sollen, müssen zuvor händisch kopiert werden.

Das Programm bietet keine Versionskontrolle an, dieser Umstand kann jedoch durch Einbinden von Zusatzwerkzeugen behoben werden.

5.2.3 Ergonomie

a) Benutzeroberfläche

Benutzeroberfläche, S. 47

1.	+
2.	+
3.	+
4.	+
5.	+
6.	0
7.	+

Die Benutzeroberfläche von objectiF ist ausschließlich aus Standard-Steuerelementen der Windows-Oberfläche aufgebaut. Das aus Windows 95 bekannte Tree-Control nimmt eine zentrale Rolle ein. Alle Browser bestehen aus einem solchen Control, in dem das Projekt über Kontextmenüs manipuliert werden kann.

Ein Umstand, der dem Benutzer sofort ins Auge sticht, ist das fehlende Hauptrahmenfenster des Programms. Nach dem Start wird nur ein kleines Fenster auf dem Desktop angelegt, das eine Menüleiste und fünf graphische

Buttons enthält. Hinter den Buttons verbergen sich Browser für verschiedene Modellaspekte. Das Programm bietet in diesem kleinen Fenster Tooltips an, d.h. fährt der Benutzer mit dem Mauszeiger über einen Button, wird in einem kleinen, einzeiligen Fenster neben der Maus eine textuelle Beschreibung der Funktionsweise des Buttons angezeigt. Dies ist nicht unwichtig, da die graphischen Buttons im Hauptfenster, zwar ansprechend gestaltet sind, jedoch trotzdem nicht immer sofort den Zweck des Buttons eröffnen. Seltsamerweise funktionieren die Tooltips nicht mehr, nachdem man einen Dialog (z.B. Modell anlegen) bedient hat.

Die Diagramme stellt das Programm in eigenen Fenstern dar, die, wie alle anderen Fenster des Programms auch, frei auf dem Desktop plaziert werden können. Die Diagrammfenster besitzen keine Menüleiste, jedoch eine links angebrachte Werkzeugleiste, deren graphische Buttons, wie im Hauptfenster, zusätzlich durch Tooltips beschrieben sind.

Die Datenbasis des Programms wird in Browserfenstern dargestellt, welche als Treeview implementiert sind. Attribute, Methoden und ganze Klassen lassen sich im Browser per Drag&Drop bewegen und anderen Elementen zuordnen. Dies funktioniert allerdings nur innerhalb eines Browserfensters, d.h. es ist nicht möglich, eine Klasse aus dem Klassenbrowser in ein Klassendiagramm zu ziehen. Um in dem Diagramm die schon bestehende Klasse einzuzeichnen, muß ein eigener Button gedrückt werden, der in einen Auswahldialog führt, in dem eine bestehende Klasse gewählt werden kann.

Die Steuerung des Programms wird in jeder Ansicht außer dem Hauptfenster, hauptsächlich von Popup-Menüs übernommen. Diese sind kontextabhängig und dynamisch generiert. Zum Beispiel können Sequenzdiagramme Klassendiagrammen zugeordnet werden. Aus dem Klassendiagramm kann dann zu einem zugehörigen Sequenzdiagramm navigiert werden. Dies geschieht über das Kontextmenü im Klassendiagramm, das im Untermenü *Sequenzdiagramme* die Namen aller zugeordneten Sequenzdiagramme als Links zu deren Ansicht bereitstellt.

Controls, die Aktionen auslösen, die im gegenwärtigen Zustand des Programms keinen Sinn ergeben, sind durchweg deaktiviert und ausgegraut. Längere Vorgänge werden durch einen entsprechenden Mauszeiger angezeigt.

Die Einträge in den Menüs des Programms sind relativ lang, sie beschreiben ihre Funktionsweise dadurch recht gut. Auch die Dialoge, durch die der Anwender mit dem Werkzeug kommuniziert sind zwar schlicht, jedoch einfach und verständlich geraten.

Zu bemängeln ist, daß objectiF kaum Tastatur-Shortcuts unterstützt. In den seltensten Fällen ist durch die Tastatur eine Reaktion auszulösen, die windowstypischen Kürzel sucht man ohnehin vergeblich. Wenngleich Cut, Copy und Paste bei objectiF wenig Sinn machen, sollte die Delete-Taste jedoch zum Löschen herangezogen werden können.

Tool-Dokumentation, S. 48

1.	+
2.	+
3.	*0*
4.	+
5.	+
6.	*0*
7.	-
8.	+
9.	*0*
10.	+
11.	+
12.	+

b) Tool-Dokumentation

Das Programm wird mit verschiedenen Online-Dokumenten im Portable-Document-Format (*.pdf) ausgeliefert. Dies sind eine einführende Quicktour, eine Expertentour, die unter anderem auf die Codegenerierung eingeht, und eine Beschreibung des Code-Scripting mit objectiF.

Weitere Informationen erhält der Benutzer durch eine Readme-Datei, die neueste Informationen und eine Beschreibung der mitgelieferten Dokumente enthält. Eine kurze Installationsanleitung ist ebenfalls in dieser Datei enthalten. Zusätzlich zu den oben genannten Dateien erhält der Benutzer ein gedrucktes Prozeßhandbuch.

c) Usability

Usability, S. 48

1.	+
2.	0
3.	+
4.	-
5.	+
6.	+
7.	0
8.	-
9.	+
10.	+
11.	0

Die Tastaturkürzel, die vom Programm zur Verfügung gestellt werden, sind, wie an anderer Stelle bereits beschrieben, äußerst dürftig. Die diesbezügliche Funktionalität von objectiF beschränkt sich darauf, daß der Benutzer die Möglichkeit hat, Menüeinträge über die gedrückte Alt-Taste zusammen mit einem Buchstaben aus dem Menübezeichner auszuwählen. Echte Shortcuts, welche die Produktivität steigern könnten, indem sie oft benutzte Befehle oder Befehlsfolgen beschleunigen, sind nicht implementiert. Das Konzept der Kontextmenüs ist in objectiF sehr gut umgesetzt. Es wird zu jedem angeklickten Objekt ein individuelles Menü angeboten.

Dem Programm eine Fehlermeldung zu entlocken gestaltet sich sehr schwierig. Zum einen ist dieser Umstand der Tatsache zuzuschreiben, daß das Programm versucht, fehlerhafte oder unsinnige Eingaben nicht zuzulassen. Dialogfelder, die beim Anlegen von neuen Elementen deren Parameter erfassen, lassen sich erst dann positiv beenden, wenn die eingegebenen Informationen vom System für gut befunden wurden. Zum anderen dürften einige Fehler bei der Entwicklung des Werkzeuges einfach übersehen worden sein:

In einem Versuch wurde eine im System vorhandene Codedatei (test.cpp) gelöscht. Der Befehl *Rückführen*, der die Datei analysieren und eventuelle Änderungen in das Projekt aufnehmen sollte, konnte ohne Fehlermeldung durchgeführt werden, obwohl die Datei nicht mehr existierte.

Es gibt sowohl im Hauptfenster des Programms als auch in den Diagrammansichten Werkzeugleisten, welche die gängigsten Befehle zur Verfügung stellen. Eine Anpassung derselben an die individuellen Bedürfnisse des Benutzers ist nicht vorgesehen. Leider bietet das Programm keine zusätzlichen Informationen an, wenn zum Beispiel im Zuge einer Modellierungsoperation mehrere Arbeitsschritte hintereinander notwendig sind. Bei vielen anderen Windowsprogrammen ist für solche Informationen eine Statuszeile am unteren Fensterrand eingerichtet.

Bei der Arbeit mit objectiF kann man beliebig viele Fenster mit verschiedenen Browser- und Diagrammansichten des Modells öffnen. Diese sind frei auf dem Desktop positionierbare Kindfenster des Hauptfensters.

objectiF ist in den Sprachen Deutsch und Englisch erhältlich, wobei die Dokumentation in der jeweiligen Landessprache verfaßt ist.

d) Hilfefunktionen

Hilfefunktionen, S. 49

1.	+
2.	+
3.	+
4.	0

Um den User bei der Nutzung des Programmes zu unterstützen, liefert Microtool mit objectiF sowohl Windows-Hilfedateien, als auch eine Dokumentation im *.pdf-Format aus. Die Windows-Hilfedateien können, wie es dem Standard entspricht, aus dem Programm heraus aufgerufen werden. Die Kontextabhängigkeit des Hilfesystems beschränkt sich darauf, daß die meisten Dialoge mit einem Hilfe-Button ausgestattet sind, welcher die Hilfedatei öffnet und die zum Thema gehörende Seite aufschlägt.

In der Hilfedatei kann nach Stichworten gesucht werden und sie besitzt ein Inhaltsverzeichnis.

Weiters stellt das Programm einen Hot-Help Mechanismus bereit. Bewegt man den Mauszeiger über eine Schaltfläche, wird in einem kleinen Fenster in der Nähe des Zeigers ein kurzer, erklärender Text zur Funktion angezeigt. Leider funktioniert dieser Mechanismus nicht so recht. Nach kurzem Arbeiten mit dem Programm und einigen Aktionen wie Öffnen und Schließen von Fenstern, zeigt das Programm diesen Hilfetext nicht mehr an. Nach einem erneuten Start von objectiF funktioniert die Hot-Help wieder.

e) Graphikeditor / Darstellung

Graphikeditor / Darstellung, S. 49

1.	-
2.	-
3.	+
4.	-
5.	-
6.	+
7.	-
8.	-
9.	+
10.	-
11.	0
12.	-
13.	+
14.	-
15.	-

Der Graphikeditor von objectiF ist in seinem Gebrauchswert stark auf die im Programm vorgesehenen Modellelemente beschränkt. Er erlaubt nicht, freie Objekte ohne definierte Bedeutung zu erstellen, oder die Darstellung der vorhandenen Modellelemente zu verändern. Zusammen mit dem Umstand, daß das Programm nur die grundlegenden UML-Konstrukte unterstützt, ergibt sich für den Benutzer eventuell die Situation, daß gewisse Teile des Modells nicht UML-konform dargestellt werden können.

Die Notationselemente jedoch, die das Programm darstellen kann, sind durchwegs gut gelungen. Die Darstellung erfolgt mehrfarbig, wobei die Farben nicht selbst gewählt werden können. Alle Elemente eines Diagramms sind durch ausgefüllte geometrische Figuren dargestellt, was die Übersichtlichkeit steigert.

Der Graphikeditor bietet nicht die Möglichkeit, Zeichnungselemente zu rotieren. Genausowenig ist es möglich, die Schriftart oder Schriftgröße zu verändern. Auch die Strichstärke der Beziehungskanten und die Farben der verschiedenen Notationselemente sind fest vorgegeben.

An Hilfsfunktionen für ein ansprechendes Diagrammlayout stellt das Werkzeug nur eine Funktion zum automatischen Anordnen der Beziehungskanten zur Verfügung. Die Anordnung der Klassen selbst muß der Benutzer händisch vornehmen. Ebenso fehlen Funktionen, die mehrere Modellelemente nach bestimmten Kriterien anordnen (*Alignment*). Die händische Layoutoptimierung wird jedoch dadurch unterstützt, daß man auch nachträglich in Beziehungskanten Eckpunkte einfügen, und eventuell entstehende Überschneidungen auflösen kann.

Die Ansicht im Diagrammfenster kann fast unbegrenzt vergrößert oder verkleinert werden, der Zoomfaktor kann jedoch nicht stufenlos verändert werden. Auch die Symbole für Klassen können in der Größe variiert werden. Dabei ist die minimale Größe die Voreinstellung und diese wird nur vom Benutzer selbst verändert und nicht z.B. durch einen überlangen Namen des Elements.

Es ist keine Möglichkeit vorhanden, den Graphikeditor zu einem neuen Fensteraufbau zu zwingen (*Refresh*). Es traten während der Tests mit dem Programm jedoch keine Situationen auf, die dies erforderlich gemacht hätten.

Der Ausdruck der Diagramme entspricht der Bildschirmansicht, es gibt eine Vorschau für das Seitenlayout und dazu eine Zentrierungs- und eine Skalierungsfunktion. Ebenfalls besteht die Möglichkeit, Schnittkanten, Klebekanten und Seitenzahlen einzufügen, für den Fall, daß ein Diagramm auf einer Druckseite nicht genug Platz findet. Auch das mehrseitige Layout wird in der Druckvorschau dargestellt.

f) Graphikeditor / Bedienung

Von der Bedienung her kann der Graphikeditor als "intuitiv zu bedienen" bezeichnet werden, jeder, der schon mit einem einfachen Zeichenprogramm unter Windows gearbeitet hat, findet sich sofort zurecht. Aus der Werkzeugleiste am linken Rand des Fensters wählt man das Element aus, das gezeichnet werden soll, wodurch der Mauszeiger eine dem Element entsprechende Form annimmt. Jetzt wird durch einfaches oder mehrfaches Linksklicken auf der Zeichenfläche das Element angelegt. Durch einen Druck auf die rechte Maustaste oder die Escape-Taste wird die Zeichenoperation unterbrochen, der Mauszeiger verwandelt sich wieder in den Windows-typischen Pfeil.

Sofort nach dem Anlegen eines neuen Elementes wird dessen Eigenschaften-Dialog geöffnet, den man nur dann vorwärts verlassen kann (*OK*), wenn man zumindest die wichtigsten Parameter eingegeben hat. Wurde ein Objekt angelegt, bleibt das Programm auch weiter im Anlegemodus für das entsprechende Notationselement, solange bis man diesen, wie zuvor beschrieben, explizit beendet. Dies ist sinnvoll, wenn mehrere Elemente desselben Typs angelegt werden sollen.

Die im Diagramm bereits vorhandenen Notationselemente können per Drag&Drop beliebig auf der Zeichenfläche verschoben werden. Obwohl der Editor nicht angewiesen werden kann, die Elemente des Diagramms auf einem Raster auszurichten, geschieht dies beim Anlegen und Verschieben derselben ganz automatisch, da die Positionierung auf der Zeichenfläche offensichtlich durch grob aufgelöste Koordinaten erfolgt.

Es können im Editor mehrere Objekte gleichzeitig gewählt und manipuliert werden. Dazu müssen diese entweder nacheinander bei gedrückter Shift-Taste ausgewählt werden, oder man zieht mit der Maus einen Rahmen um sie.

Der Graphikeditor bietet leider keine Undo-Funktion. Auch Cut, Copy und Paste Funktionalität sucht man vergeblich, obwohl diese im konkreten Zusammenhang weniger sinnvoll erscheint. Der Umstand, daß man den Löschvorgang nicht mit der Entfernen-Taste eingeleitet werden kann, sondern daß dazu ein Eintrag aus dem Kontextmenü gewählt werden muß, erscheint unlogisch und unnötig.

Das Zeichenfenster besitzt für beide Achsen Scrollbalken an den Rändern und ist im wahrsten Sinne des Wortes unendlich groß. Wenn man nämlich an den vermeintlichen unteren oder rechten Rand des Blattes scrollt, läßt sich die Ansicht über die Pfeiltasten dennoch weiterscrollen. Die Positionsindikatoren auf dem Scrollbalken bewegen sich hierbei nicht. Scrollt man später in die entgegengesetzte Richtung, bewegt sich der Indikator zu Beginn des Scrollvorganges nicht, er funktioniert erst wieder, wenn man in den definierten Bereich des Blattes gelangt.

Graphikeditor / Bedienung, S. 50

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	+
8.	+
9.	+
10.	-
11.	-
12.	+
13.	-
14.	-

g) Performance

Die Geschwindigkeit mit der objectiF zu Werke geht kann als sehr gut beurteilt werden. In keiner Phase der Tests mit dem Werkzeug mußten längere Wartezeiten in kauf genommen werden. Es ist allerdings zu berücksichtigen, daß objectiF eines der einfacheren Programme des Tests war, wobei hier "einfacher" nicht mit "schlechter" gleichgesetzt werden darf. Es verzichtet im Vergleich zu den komplexeren Produkten auf einige Funktionalität, wodurch sicherlich seine Geschwindigkeit profitiert.

Performance, S. 51

1.	+
2.	+
3.	*0*
4.	+

h) Adaptivität

In Bezug auf Adaptivität hat objectiF dem Benutzer wenig zu bieten. Menüs und Werkzeugleisten lassen sich nicht an die individuellen Bedürfnisse anpassen und es existiert auch keine Möglichkeit, um Makros aufzuzeichnen. Um Dokumentationen automatisch generieren zu lassen, besteht die Möglichkeit, COM-fähige Programme als COM-Client mit objectiF kommunizieren zu lassen. Dadurch ist es möglich, mit MS-Word individuelle Dokumentationen generieren zu lassen, die auf den Daten des Projekts basieren.

objectiF wird nicht im Sourcecode ausgeliefert.

Adaptivität, S. 51

1.	-
2.	-
3.	-
4.	-
5.	+
6.	-

5.3 Objekttechnologie-Werkbank 2

Die Objekttechnologie-Werkbank 2 ist ein Produkt der deutschen Firma OWiS Software GmbH aus Illmenau. Das Produkt ist für die Windows-Plattform programmiert und ist sowohl unter Windows 9x, als auch unter Windows NT ab Version 3.52 lauffähig.

OTW-2 stellt dem Benutzer das sogenannte Objekt-Prozeß-Modell (OPM) zur Verfügung, das nicht UML-konform ist. Dieses ist eine Notation für die objektorientierte Prozeßmodellierung und wurde in [Burk94] definiert. Es ist als Alternative zum Zustandsmodell (Zustands- oder Aktivitätsdiagramm) und als Ergänzung zum Interaktionsmodell (Sequenz- oder Kollaborationsdiagramm) zu sehen.

Das Interaktionsmodell beschreibt den konkreten Ablauf einer Methode in einem bestimmten Einzelfall. Das Objekt-Prozeß-Modell hingegen erlaubt die Modellierung von Vor- und Nachbedingungen für Methoden, die unabhängig vom jeweiligen Aufruf sind und immer gelten.

Die im OPM-Schema gemachten Angaben können bei der Codegenerierung direkt in entsprechenden Überprüfungscode überführt werden und erhöhen so die Sicherheit des Zielsystems (vgl. [Burk97]).

OTW-2 ist als Standard-, Team-, Entwickler-, sowie als nichtkommerzielle Privat-Edition erhältlich.

5.3.1 Methodenbasierte Kriterien

a) Modellierung der Objektstruktur - Objektstatik

Modellierung der Objektstruktur - Objektstatik, S. 38

1.	+
2.	+
3.	-
4.	+
5.	+
6.	+
7.	*0*

OTW-2 verwaltet die Daten eines Projektes in einem sogenannten Repository, welches auf einem objektorientierten Datenbanksystem basiert. Das Programm lädt nach dem Start das Repository des zuletzt bearbeiteten Projektes. Soll ein anderes Projekt geladen werden, geschieht dies über den Dialog *Repositorys verwalten*, in dem das Programm dem Benutzer alle bekannten Projekte zur Auswahl gibt. Welche physikalischen Dateien hinter den Projekten stehen, bleibt dem Benutzer hier verborgen. Diese Informationen sind jedoch über den Menüeintrag *Repository-Informationen* abrufbar.

Das Werkzeug bietet eine als Tree-View ausgeführte Paketübersicht, in welcher der Inhalt des geladenen Repositorys angezeigt wird. Hier werden alle Pakete mit den darin gespeicherten Notationselementen, Diagrammen, eventuellen Unterpaketen usw. angezeigt. Dieser Browser kann zum Navigieren im System oder zum direkten Manipulieren von Modellelementen benützt werden.

Weiters existiert eine Diagrammübersicht, in der die zu dem Projekt vorhandenen Diagramme und deren Bestandteile, in einer Tree-View dargestellt werden. Diese Übersicht bietet, in Bezug auf Navigation und Manipulation, die selbe Funktionalität wie die oben erwähnte Paketübersicht.

Bestehende Elemente können auch nachträglich über deren Eigenschaften-Dialog bearbeitet werden. Hierbei können alle Parameter verändert werden, eventuelle Inkonsistenzen, wie in einem Namensraum doppelt vergebene Bezeichner, werden vom Programm erkannt und verhindert.

OTW-2 erlaubt es, zu jedem Attribut einen Datentyp anzugeben. Methoden erhalten einen Rückgabetyp und eine Parameterliste, wobei jeder Parameter wieder einen Datentyp erhält. Die Reihenfolge der Parameter innerhalb der Parameterliste einer Methode kann auch im Nachhinein verändert

werden. Zu jedem Parameter kann weiters angegeben werden, ob dieser ein Wert, ein Zeiger oder eine Referenz ist, und ob dieser in der Methode verändert werden darf oder nicht (*konstant*). Eine definitive Einschränkung, ob es sich bei dem Parameter um einen Eingangs-, Ausgangs- oder Durchgangsparameter handelt, kann der Benutzer jedoch nicht vornehmen.

Die Reihenfolge der Darstellung der Komponenten einer Klasse im Diagramm kann vom Benutzer nicht nachträglich verändert werden. Hier ist ausschlaggebend, welche Komponente zuerst angelegt wurde.

Vor- und Nachbedingungen für Methoden lassen sich mit dem in OTW-2 enthaltenen Objekt-Prozeß-Modell erstellen. Dieses Modell ist ein Zusatz zur UML und wurde für die Modellierung dynamischer Aspekte entwickelt.

b) Modellierung der Systemstruktur - Systemstatik

Modellierung der Systemstruktur - Systemstatik, S. 39

1.	+
2.	+
3.	-
4.	-

In OTW-2 können rekursive Aggregationsbeziehungen angelegt werden. Die Darstellung einer solchen Konstruktion im Klassendiagramm muß jedoch als unschön bezeichnet werden. Die vom Klassensymbol wegführende und die zum Klassensymbol hinführende Teilkante überlagern sich und es entsteht der Eindruck einer Kante, die ins Leere führt. Es besteht allerdings die Möglichkeit, die Kante manuell so anzuordnen, daß eine Schlaufe entsteht. Dazu muß über das Kontextmenü ein zusätzlicher Knickpunkt in die Kante eingefügt, und diese dann per Drag & Drop justiert werden.

Beziehungen werden in OTW-2 über denselben Dialog angelegt wie Attribute. Man kann als Beziehungsattribut ein bestehendes Attribut der Klasse auswählen, oder ein neues anlegen. Weiters sind die angelegten Beziehungen nur in eine Richtung navigierbar. Will man eine beidseitig navigierbare Assoziation, muß man zwei gegengleiche Rollenbeziehungen anlegen. Die angelegten Relationen können auch nachträglich über den Eigenschaften-Dialog des Beziehungsattributes manipuliert werden, es ist jedoch unmöglich, eine der miteinander in Beziehung stehenden Klassen durch eine andere zu ersetzen.

Außer der Zuordnung der Klassen in Module und Packages ist keine Strukturierung des Projektes vorgesehen. OTW-2 unterstützt keine OCL-Annotation.

c) Dynamisches Objektverhalten - Objektdynamik

Dynamisches Objektverhalten - Objektdynamik, S. 39

1.	-
2.	-
3.	-
4.	-
5.	-
6.	-

Die OTW-2 unterstützt keine Zustandsdiagramme, wie sie in der UML vorgesehen sind. Alternativ wird, wie in der Einleitung erwähnt, das sogenannte Objekt-Prozeß-Modell (OPM) angeboten.

OPM-Diagramme basieren nicht auf Zuständen, sondern auf Objekten und Prozessen, als konkrete Ausprägungen von Klassen bzw. deren Methoden. Wie in Zustandsdiagrammen existieren in OPM-Diagrammen Kanten, die Übergänge zwischen den Diagrammelementen darstellen. Diese Übergänge müssen mit mindestens einer Bedingung versehen werden, wobei auf die Attribute des Objektes Bezug genommen wird.

Allen Diagrammelementen außer Kanten kann eine textuelle Beschreibung beigefügt werden.

d) Dynamisches Systemverhalten - Systemdynamik

Die Aggregation von Zuständen nebenläufiger Objekte kann nicht modelliert werden.

Das Sequenzdiagramm ist grafisch sehr gut umgesetzt. Konstruktion und Destruktion von Objekten ist, im Gegensatz zu einigen Mitbewerbern, der UML-Notation entsprechend machbar.

Beim Sequenzdiagramm muß auf das statische Modell Bezug genommen werden. Für jede Botschaft, die in diesem Diagramm angelegt wird, muß eine korrespondierende Methode in dem Empfängerobjekt existieren. Ist dies nicht der Fall, wird eine neue Methode in der entsprechenden Klasse angelegt. Im Sequenzdiagramm gelangt man in den Eigenschaften-Dialog einer Methode, über den man alle ihre Merkmale wie Rückgabetyp, Parameter, Sichtbarkeit usw. verändern kann. Eventuelle Veränderungen wirken sich auf das statische Modell aus.

Dynamisches Systemverhalten - Systemdynamik, S. 39

1.	-
2.	+

e) Zusammenhang zwischen den Modellen

Jedes OPM-Diagramm ist eindeutig einem Objekt, einer Klasse oder einem Prozeß zugeordnet. OTW-2 bietet dem Benutzer auch die Möglichkeit, einfach zwischen zusammengehörigen Sichten eines Modellelementes zu navigieren. Dies geschieht über einen Eintrag im Kontextmenü eines Elementes oder durch einen Doppelklick darauf.

Bei der Modellierung des dynamischen Modells kann auf Attribute und Methoden des statischen Modells Bezug genommen werden.

Zusammenhang zwischen den Modellen, S. 39

1.	-
2.	-
3.	+
4.	+

f) Abstraktionsebenen / Sichten

Der Detaillierungsgrad von Klassendiagrammen läßt sich über den Dialog *Darstellung* verändern. Dies kann auf drei Ebenen geschehen: Zuerst können Grundeinstellungen vorgenommen werden, die für alle Projekte gelten. Auf der nächsten Ebene können Voreinstellungen für ein konkretes Projekt vorgenommen werden und zuletzt kann die Darstellung jedes einzelnen Diagramms variiert werden. Der Umfang der Einstellmöglichkeiten ist enorm. Für jede Diagrammart kann die Darstellung der darin vorkommenden Notationselemente in Bezug auf Farbe, Schriftart, Schriftgröße, textuelle Beschreibung usw. verändert werden. Weiters können Stereotype, Methoden und Attribute ein- und ausgeblendet werden.

Ein weiteres sehr nützliches Feature ist, daß über den *Darstellung* Dialog die Standardmethoden einer Klasse ein- bzw. ausgeblendet werden können. Zu diesen gehören Konstruktor, Destruktor, Copy-Konstruktor und Zuweisungsoperator. Die Einstellungen werden in jedem Fall über denselben, sehr übersichtlichen Dialog abgewickelt.

Bei der ganzen Fülle von Einstellmöglichkeiten gibt es leider immer noch einige Anpassungen, die eventuell nützlich sein könnten, jedoch nicht vorgenommen werden können: Die Darstellung der Signatur von Attributen und Methoden läßt sich nicht verändern. Attribute werden immer mit ihrem Typ, Methoden mit Rückgabetyp und Parameterliste angegeben. Weiters besteht keine Möglichkeit, die Darstellung auf Modellelemente zu beschränken, die gewisse individuell festzulegende Bedingungen (z.B. Sichtbarkeit *public*) erfüllen.

Im Optionen-Menü kann die Darstellungsart verändert werden, wobei außer der UML-Darstellung auch die Notation von Brachman [Brac95] zur Verfügung steht.

Abstraktionsebenen / Sichten, S. 40

1.	+
2.	-
3.	-
4.	-
5.	+

g) Vorgangsreihenfolge bei der Entwicklung

Die Vorgehensweise beim Modellieren eines Projektes bestimmt der Benutzer. Wird ein neues Modellelement angelegt, gelangt man sofort in einen

umfangreichen Eigenschaften-Dialog, es muß zunächst jedoch nur der Name des Elementes angegeben werden. Dieser kann jedoch im nachhinein nicht mehr geändert werden. Ist eine Änderung erforderlich, muß das Element gelöscht und neu angelegt werden. Zu beachten ist auch, daß beim Anlegen einer neuen Klasse deren Typ festgelegt wird. Zur Auswahl stehen Klasse, Template und Template-Instanz, wobei *Klasse* voreingestellt ist. Dieser Typ der Klasse kann im nachhinein, wie der Name, nicht verändert werden.

Vorgangsreihenfolge bei der Entwicklung, S. 40

1.	*0*
2.	*+*
3.	*+*
4.	*-*

Es ist möglich, daß Modellelemente bereits im Data-Dictionary gespeichert sind, obwohl sie in keinem Diagramm vorkommen. Eine solche Situation kann entstehen, wenn eine Quellcodedatei eingelesen wurde, oder wenn Modellelemente aus einem Diagramm gelöscht wurden, ohne sie aus der Datenbasis zu entfernen. Solche Elemente können ohne Schwierigkeiten in Diagramme eingetragen werden.

Die Möglichkeit, Eingaben händisch im Repository vorzunehmen, oder existierende Modellelemente außerhalb eines Diagrammes zu editieren, besteht im Paketübersicht-Browser.

Alle Modellelemente, die im System angelegt werden, werden sofort im Repository gespeichert. Dabei ist es egal, in welchem Diagramm sie angelegt werden.

h) Unterstützung bei der Methodenanwendung

Die Objekttechnologie-Werkbank 2 ist kein reines UML-Werkzeug. Es geht in vieler Hinsicht eigene Wege, wie dies am Objekt-Prozeß-Modell sichtbar wird.

Unterstützung bei der Methodenanwendung, S. 40

1.	*-*
2.	*-*
3.	*-*
4.	*0*
5.	*-*
6.	*-*

Das mitgelieferte Tutorial beschreibt genau die Entstehung eines Projektes mit Hilfe der OTW-2. Dabei wird aber weniger auf die UML eingegangen, als auf die Benutzung des Programmes. Das Zustandsdiagramm, welches im Programm durch das OPM-Diagramm ersetzt wurde, wird nicht erwähnt. Eine wirkliche UML-Hilfe oder ein UML-Tutorial gibt es nicht.

Das Werkzeug basiert auf UML 1.1, die Konzepte, die von OTW umgesetzt werden, werden im großen und ganzen auch UML-konform dargestellt.

5.3.2 Funktionale Kriterien

a) Datenhaltung

Die Daten werden in der OTW-2 in einem zentralen Repository gespeichert, wobei jedes Projekt ein separates Repository erhält. Die Repositorys basieren auf einem objektorientierten Datenbanksystem. Konkret ist dies eine modifizierte POET-Datenbank. Konsistenz und Zugriff werden von OTW selbst geregelt

Datenhaltung, S. 41

1.	*+*
2.	*+*
3.	*+*
4.	*+*
5.	*-*
6.	*+*
7.	*-*
8.	*-*

In dieser Datenbasis werden die Daten aller Teilmodelle verwaltet, mehrfach vorkommende Elemente werden nicht redundant gespeichert. Im Repository vorkommende Modellelemente müssen nicht zwingend in einem Diagramm vorkommen. Beispielsweise werden beim Reverse Engineering die Daten aus dem gegebenen Quellcode extrahiert und in die Datenbasis übernommen. Erst in einem zweiten Schritt werden diese Daten dann in ein Diagramm eingetragen.

Weiters kann das Datenformat nicht als offen bezeichnet werden. Es gibt keine Im- oder Exportfunktion, die auf der Basis von ASCII-Dateien funktioniert und auch keine Schnittstellen zu handelsüblichen relationalen Datenbanksystemen.

b) Browsing-Eigenschaften

Browsing-Eigenschaften, S. 41

1.	-
2.	-
3.	0
4.	-
5.	-
6.	-
7.	+
8.	0
9.	+
10.	-
11.	-

Die Möglichkeiten, die OTW-2 dem Benutzer zur Verfügung stellt, um Modellelemente aufzufinden, sind sehr beschränkt. Sowohl im Paketübersicht- als auch im Diagrammübersicht-Browser wird bei Eingabe eines Zeichens über die Tastatur der Fokus auf das nächste Element mit passendem Anfangsbuchstaben gesetzt. Eine richtige Suchfunktion bietet das Programm nicht. Positiv ist, daß in diesen beiden Browserfenstern auch Veränderungen an Modellierungsinformationen durchgeführt werden können.

Der Benutzer hat die Möglichkeit, sich zu der gewählten Klasse deren Super- und Subklassen anzeigen zu lassen. Das geschieht über einen Dialog mit zwei Listenfeldern. Es ist unmöglich, in einer Klassenhierachie von einer Klasse zu einer anderen zu navigieren.

OTW-2 bietet keine statistischen Informationen über einzelne Modellelemente an. Lediglich über das gesamte Repository lassen sich Informationen abrufen.

Elemente, die zwar im Repository, jedoch noch in keinem Diagramm vorkommen, werden genauso ausgewählt wie alle anderen. Eine Funktion, die gerade solche Elemente anzeigt oder hervorhebt, existiert nicht.

c) Navigation im Systemmodell

Navigation im Systemmodell, S. 42

1.	+
2.	+
3.	-
4.	+

Das Programm erlaubt dem Benutzer über Kontextmenüs im Systemmodell zu navigieren. Es können so Verfeinerungen im Objekt-Prozeß-Modell zu einer Klasse angezeigt werden, diese Klasse im Use-Case- bzw. Klassendiagramm dargestellt werden, die Komponente des gewählten Modellelementes im Komponentendiagramm dargestellt werden, usw. Wird ein Element im Diagramm angewählt und mit der rechten Maustaste das Kontextmenü geöffnet, sind nur die tatsächlich nutzbaren Navigationsmöglichkeiten aktiviert, alle anderen werden ausgegraut dargestellt.

d) Konsistenzprüfung / Integrität

Konsistenzprüfung / Integrität, S. 42

1.	+
2.	+
3.	+
4.	+
5.	+
6.	-
7.	-
8.	-
9.	+

Die Objekttechnologie-Werkbank führt eine Konsistenzprüfung schon beim Anlegen eines Diagramms durch, ein vom Benutzer explizit aufrufbarer Konsistenz-Check ist nicht vorgesehen. Es ist nicht möglich, unbenannte Elemente zu erzeugen, stattdessen muß ein neuer Name angegeben oder ein bereits vorhandenes Element aus dem Repository ausgewählt werden. Weiters erlaubt es das Programm nicht, daß Beziehungen angelegt werden, die ins Leere gehen, oder Vererbungsbeziehungen, bei denen Super- und Subklasse ident sind (rekursive Vererbung). Wird eine Klasse mit einem Namen angelegt, der schon an eine andere Klasse im selben Paket vergeben wurde, wird eine Warnung ausgegeben und die Eingabe ignoriert.

Um ein Modellelement aus einem Diagramm zu löschen, stehen dem Benutzer zwei Befehle zur Verfügung: Er kann das Element aus dem Diagramm löschen, wodurch das Repository nicht verändert wird. Die zweite Möglichkeit besteht darin, das Modellelement komplett aus dem Repository und damit aus dem gesamten Projekt zu entfernen. Die zweite Variante wirkt sich zwangsweise auf alle Diagramme des Modells aus, in denen das betreffende Element vorkommt.

e) Report-Ausgabe

Eine Dokumentation zu einem Projekt wird mit der OTW-2 in zwei Schritten erstellt. Zuerst wird die Dokumentation mit dem gewünschten Umfang und

Detaillierungsgrad erstellt. Dabei legt das Programm eine ASCII-Textdatei an, welche die gewünschten Daten enthält. Als nächstes werden diese Daten mittels eines geeigneten Automationsmechanismus in das gewünschte Ausgabeformat gebracht. Hierbei können auch die in OTW-2 angelegten Diagramme verwendet werden, da sie vom Programm als Bitmap gespeichert werden, und die Pfade und Bezeichnungen dieser Dateien im ASCII-Dokumentationsfile gespeichert sind.

Report-Ausgabe, S. 43

1.	+
2.	+
3.	+
4.	+
5.	+
6.	-
7.	-
8.	-
9.	+
10.	+

Der Mechanismus der Dokumentgeneration bietet umfassende Anpassungsmöglichkeiten. Im Einstellungs-Dialog kann der Benutzer 65 verschiedene Parameter verändern, die bestimmen, was in die Dokumentation aufgenommen wird und was nicht. Der Hersteller liefert schon einige Standardschablonen für öfters benötigte Dokumentationen wie zum Beispiel Klassendokumentation oder Diagrammdokumentation mit. Es lassen sich jedoch auch auf einfachem Weg Dokumentationsschablonen selbst anlegen, oder vorhandene Schablonen anpassen.

Außer den Informationen, die OTW-2 generiert, werden auch Kommentare in die Dokumentation aufgenommen, die der Benutzer jedem Element zuweisen kann. Weiters besteht die Möglichkeit, jedem Modellelement eine externe Dokumentationsdatei zuzuordnen, die dann ebenfalls in die Dokumentation aufgenommen wird. Solche Dateien müssen entweder im MS-Word-Format (*.doc) sein oder die Extension *txt* besitzen und ASCII-Text enthalten.

Die Dokumentgeneration mit MS-Word verläuft weitgehend problemlos und kann als optisch sehr ansprechend bezeichnet werden. Ein solches Dokument besitzt eine Titelseite, ein Inhaltsverzeichnis, Kopf- und Fußzeile und beinhaltet, sofern Diagramme ausgegeben werden, farbige Grafiken.

f) Datenbankschema-Ausgabe

Datenbankschema-Ausgabe, S. 43

1.	+
2.	-
3.	-
4.	-
5.	-
6.	-

Um die Anbindung an ein relationales Datenbanksystem zu ermöglichen, stellt die OTW-2 einen Generator zur Verfügung, der eine DDL-Datei (Data Definition Language) nach dem SQL-92 Standard erstellt. Die Generierung des DDL-Codes erfolgt modulbezogen. Für jedes Modul wird eine DDL-Datei angelegt. Innerhalb Datei befinden sich für alle Klassen des Moduls die DDL-Statements. Der Benutzer kann über eine Konfigurationsdatei die Zuordnung von domänenspezifischen Datentypen (Klassen der OTW-2) zu Datentypen der Datenbank definieren. Somit ist die Anbindung an alle relationale Datenbanken möglich.

Für jede Klasse und jedes mehrwertige Attribut wird eine Tabelle definiert. Die vollständige Abbildung einer Klasse wird letztendlich durch Views realisiert.

Es werden automatisch passende Zugriffsmethoden zu den relevanten Attributen angelegt. Für die Implementierung der Zugriffsmethoden kann der Benutzer Voreinstellung benutzen oder über eine Konfigurationsdatei eine beliebige andere Implementierung durch eine einfache Definitionssprache spezifizieren.

g) Diagramm-Ausgabe

Diagramm-Ausgabe, S. 44

1.	+
2.	-
3.	-
4.	+
5.	-
6.	-

Beim Ausdruck von Diagrammen greift OTW-2 als Windows-Programm auf die Druckerunterstützung des Betriebssystems zurück, wodurch sämtliche Drucker, die von Windows unterstützt werden, benutzt werden können. Das Programm selbst kann die Qualität des Ausdrucks nicht beeinflussen, was jedoch auch keinen Sinn machen würde, da diese Funktionalität ohnehin vom Druckertreiber zur Verfügung gestellt wird. Das gleiche gilt für die Aus-

richtung (Hochformat - Querformat) des Ausdrucks. Es existiert eine Druckvorschau, die genau mit dem resultierenden Ausdruck übereinstimmt.

Die Detaillierungsstufe des Ausdrucks läßt sich nicht explizit einstellen, jedoch kann diese indirekt über den Diagrammdarstellungs-Dialog vorgenommen werden, da der Ausdruck ein genaues Abbild der Diagrammansicht auf dem Bildschirm ist.

Bei der Diagramm-Ausgabe auf dem Drucker ist zu bemängeln, daß das Programm nur einseitige Ausdrucke zuläßt und dabei die Diagrammansicht automatisch so skaliert, daß die ganze Seite ausgefüllt ist. Hierdurch werden kleine Diagramme stark vergrößert dargestellt, was optisch nicht überzeugt. Andererseits werden sehr große Diagramme mitunter so stark verkleinert, daß Details kaum mehr zu erkennen sind. Hier wäre eine mehrseitige Ausgabe mit Kleberändern und ein vom Benutzer einstellbarer Skalierungsfaktor wünschenswert.

Die von OTW-2 produzierten Ausdrucke entsprechen optisch der Darstellung auf dem Bildschirm. Viele Modellelemente werden farbig ausgefüllt dargestellt und die Schriftart kann frei gewählt werden. Insgesamt kann die Druckerausgabe als Präsentationsfähig bezeichnet werden, wenn das auszugebende Diagramm entsprechende Dimensionen aufweist. Bei besonders großen oder kleinen Diagrammen ist das Ergebnis unbefriedigend.

h) Offenheit - Import / Export

Offenheit - Import / Export, S. 44

1.	-
2.	*0*
3.	+
4.	-
5.	+
6.	*0*
7.	*0*
8.	-

Den Autoren dieses Textes sind keine Zusatztools zu der OTW-2, weder vom Hersteller, noch von Drittanbietern, bekannt.

Der Datenaustausch ist, wie für Windowsprogramme üblich, über die Zwischenablage möglich. Hierbei ist zu beachten, daß in den Grafikeditor nichts eingefügt werden kann. Das von vielen Programmen unterstützte OLE (*Object Linking and Embedding*) wird nicht unterstützt, ebensowenig die Generierung von gängigem Hypertext-Code.

OTW-2 arbeitet sehr gut mit MS-Word zusammen, und läßt sich auch an Entwicklungsumgebungen wie MS-Visual C++ oder Borland C++ anbinden.

In das Modell lassen sich als Dokumentation zu Modellelementen externe Textdateien einbinden. Diese können ASCII-Text enthalten oder im MS-Word Format vorliegen. Der Grafikeditor erlaubt keinen Import von toolfremden Grafikformaten, lediglich Piktogramme für Stereotypen können als *.emf Dateien (*Enhanced Meta-File*) eingebunden werden.

i) Code-Generierung

Code-Generierung, S. 45

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	+
8.	+
9.	-
10.	+
11.	-

Der Codegenerator von OTW-2 kann C++ und Java-Code erzeugen. Das Programm arbeitet mit den Programmierumgebungen MS Visual C++, Borland C++ und Visual Age C++ zusammen, für welche auch überarbeitete Headerdateien mitgeliefert werden, die für den Scanner/Codegenerator benötigt werden.

Die Codeerzeugung kann nur vom Komponentendiagramm aus gestartet werden. Der Code wird für jedes Modul erzeugt und es entstehen dementsprechend einzelne Dateien. Es sind zwei Methoden verfügbar, wie der Code erzeugt werden kann: Zum ersten kann das Modul *vorwärts generiert* werden, wobei bestehender Code einfach überschrieben wird. Die zweite Möglichkeit besteht darin, den Code *inkrementell* generieren zu lassen. Hierbei erzeugt OTW-2 den Code nach der Modellvorgabe, Programmzeilen, die der Benutzer selbst in den Quelltext eingefügt hat, bleiben jedoch erhalten. Damit der Code inkrementell erzeugt werden kann, muß die Quelldatei zuvor gescannt worden sein. Es ist leider keine Möglichkeit vorgesehen, den Code

schon im Werkzeug zu erstellen. Lediglich Kommentare können im Modell eingefügt werden und sind dann auch im generierten Code als Kommentare vorhanden.

OTW-2 generiert aus dem Modell ein komplettes Programmgerüst. Zu jeder Klasse werden auf Wunsch Konstruktor, Destruktor, Copy-Konstruktor, Vergleichs- und Zuweisungsoperator erzeugt. Es werden sämtliche betroffenen Klassen, Attribute und Methoden deklariert. Daten-, Rückgabetypen und Funktionsparameter werden ebenso berücksichtigt wie Zugriffsbeschränkungen. Beziehungen zwischen Klassen werden vom Codegenerator berücksichtigt und richtig umgesetzt.

j) Reverse-Engineering

Reverse-Engineering, S. 45

1.	*+*
2.	*+*
3.	*0*

Mit OTW-2 läßt sich ohne Probleme beliebiger, nicht vom Programm selbst generierter Code einlesen. Es wird hierbei ein neues Modul angelegt, das die Quellcodedatei repräsentiert. Der einzulesende Code muß vor der Analyse nicht gesondert bearbeitet werden, es gilt nur sicherzustellen, daß der Scanner von OTW-2 Zugang zu allen im Code eingebundenen Headerdateien hat. Im Optionendialog können zu diesem Zweck Suchpfade angegeben werden.

Code, der über eine Klassen- und Funktionsdeklaration hinausgeht, wird von der OTW-2 nicht verwendet und kann während der Modellierung nicht eingesehen oder verändert werden. Solcher Code geht aber beim Scanvorgang nicht verloren. Wird die Quellcodedatei beim nächstenmal inkrementell generiert, wird zuvor beim Scannen gefundener nicht analysierter Code wieder an seinen Platz zurückgeschrieben.

Eingelesener Code wird vom Programm interpretiert und zunächst nur in das Repository aufgenommen. Diagramme, in denen Modellelemente vorkommen, die sich durch den Scanvorgang verändert haben, werden aktualisiert. Sollen neue Modellelemente in Diagrammen dargestellt werden, müssen sie dort händisch eingefügt werden. OTW-2 bietet die Möglichkeit, Relationen anzuzeigen, die noch nicht im Diagramm eingezeichnet sind.

k) Test / Verifikation

Test / Verifikation, S. 45

1.	*-*
2.	*0*

OTW-2 enthält keine Funktionen, die ein Testen von Modellen zulassen. Der Hersteller weist jedoch in der mitgelieferten Dokumentation darauf hin, daß das Objekt-Prozeß-Modell mit geeigneten Werkzeugen verifizierbar ist. Es könnte in ein gefärbtes Petri-Netz umgewandelt und so auf Konfliktfreiheit, Lebendigkeit, Erreichbarkeit und Sicherheit überprüft werden.

l) Maßzahlen und Statistiken

Maßzahlen und Statistiken, S. 46

1.	*-*
2.	*-*
3.	*-*
4.	*-*

Die Objekttechnologie-Werkbank 2 ist nicht in der Lage, Maßzahlen oder Statistiken zu einem Projekt zu berechnen.

m) Multi-User - Eigenschaften

Multi-User - Eigenschaften, S. 46

1.	*+*
2.	*-*
3.	*+*
4.	*+*
5.	*-*
6.	*-*
7.	*-*

Das Programm ist in der Team-Edition von bis zu 100 Benutzern gleichzeitig im lokalen Netzwerk ausführbar. OTW-2 besitzt eine integrierte Nutzerverwaltung, die nur von Benutzern mit Administratorrechten bedient werden kann.

Das Programm kann Nutzerrechte nur auf Paketebene vergeben. Will ein Benutzer an einem Multi-User-Projekt arbeiten, muß er einen Arbeitsbereich anlegen, für den ein dupliziertes Repository erstellt wird. Jetzt werden dem Benutzer für die von ihm gewünschten Pakete volle Zugriffsrechte zugewie-

sen, sofern diese nicht bereits anderweitig vergeben sind. Auf alle anderen Pakete des Repositorys ist dem Anwender nur Lesezugriff erlaubt. Pakete, die von Anwendern benutzt werden, werden im Master-Repository gelockt, um zu vermeiden, daß von mehrere Seiten gleichzeitig schreibend auf sie zugegriffen wird.

Wenn ein dupliziertes Repository wieder ins Master-Repository eingefügt wird, werden eventuell auftretende Inkonsistenzen automatisch beseitigt. Das Master-Repository bleibt immer konsistent.

n) Security

Security, S. 46

1.	*0*
2.	*-*
3.	*-*
4.	*+*
5.	*-*
6.	*-*
7.	*+*
8.	*-*
9.	*+*
10.	*+*
11.	*+*

In der Teamversion der OTW-2 werden in einer Benutzerdatenbank die registrierten Nutzer gehalten. OTW-2 verwendet das Benutzerkonzept von Windows um die Identität eines Anwenders zu ermitteln und prüft anhand der Nutzerdatenbank dessen Berechtigungen. Die Zugriffssicherheit sowie auch die Konsistenz der Modellinformationen werden nach dem *Checkin / Checkout*- Verfahren gesichert.

Log-Dateien, mit deren Hilfe Aktionen auf bestimmte Benutzer zurückgeführt werden können, sind in OTW-2 nicht verfügbar. Dies soll laut Hersteller jedoch mit der Business-OTW (BOTW) behoben werden, die als Aufsatz zur OTW-2 gedacht ist, und die sich in Entwicklung befindet. Die BOTW wird neben Logdateien auch Nutzerprofile mit variablem Layout enthalten.

Da das Repository des Werkzeuges auf einem Datenbanksystem aufbaut, ist für Recovery zum Beispiel nach einem Systemabsturz gesorgt.

Die Vergabe von Rechten im Multi-User Betrieb erfolgt durch den Projektleiter, der über Administratorrechte verfügen muß.

o) Multi-Projekt-Eigenschaften / Versionsmanagement

Multi-Projekt-Eigenschaften / Versionsmanagement, S. 47

1.	*+*
2.	*+*
3.	*-*
4.	*0*
5.	*-*

OTW-2 erlaubt das Anlegen mehrerer Repositorys. Diese können bei Bedarf geladen, und so mehrere Projekte parallel abgewickelt werden. Um zwischen zwei Projekten Modellierungsinformationen auszutauschen, stellt das Programm eine Funktion zum Importieren von Repositorys in das geöffnete Projekt bereit. Hiermit lassen sich jedoch nur ganze Repositorys importieren. Es ist nicht möglich, nur bestimmte Teile eines Modells in ein anderes zu übernehmen.

Komponenten von anderen Projekten zu verwenden, ohne diese zu kopieren, wird von OTW-2 nicht unterstützt.

Das Werkzeug unterstützt die Verwendung eines externen Versionierungstools. Dieses kann im Optionendialog konfiguriert werden und wird durch einen eigenen Eintrag im Repository-Menü aufgerufen. So ist es möglich, die bestehende Version des Projektes zu konservieren und bei Bedarf zu ihr zurückzukehren.

5.3.3 Ergonomie

a) Benutzeroberfläche

OTW-2 ist vom Look-And-Feel her ein typisches Windows-Programm. Das User-Interface entspricht dem aktuellen Standard, und verwendet die aktuellen Steuerelemente wie Treeviews und Kontextmenüs. Das Programm legt ein Hauptrahmenfenster an, in dem beliebig viele frei veränderbare Kindfenster Platz finden. Das Hauptrahmenfenster besitzt je eine Werkzeugleiste

am oberen und am linken Rand. Die Buttons der oberen Werkzeugleiste dienen zum Auffinden des gewünschten Kindfensters, während sich die seitliche Leiste dynamisch an das gewählte Fenster anpaßt und häufig benötigte Menübefehle als Button zur Verfügung stellt.

Benutzeroberfläche, S. 47

1.	*0*
2.	*+*
3.	*+*
4.	*+*
5.	*+*
6.	*0*
7.	*+*

Da alle Kindfenster frei plaziert und vergrößert werden können, sind häufig benötigte Fenster, wie das Übersichtsfenster, oft verdeckt und müssen mit dem entsprechenden Button im Toolbar in den Vordergrund gebracht werden. Die bessere Lösung wäre hier sicher eine geteilte Ansicht, bei der das Übersichtsfenster nie verdeckt werden kann.

Die Dialoge von OTW-2 sind übersichtlich gestaltet und machen extensiven Gebrauch vom Registerseiten-System. Die Tabulator-Reihenfolge der Kontrollelemente in den Dialogen kann als intuitiv bezeichnet werden. Die Menüleiste des Hauptrahmenfensters entspricht dem Windows-Standard, die Standard-Popupmenüs befinden sich dort, wo sie ein geübter Windows-Benutzer erwartet (links: *Datei* - *Bearbeiten* - *Ansicht*; ganz rechts: *Fenster* - *Hilfe*). Die Bezeichnungen der Menüeinträge sind lange ausgefallen, dafür aber weitgehend selbsterklärend. Das über die rechte Maustaste aufrufbare Kontextmenü ist umfangreich und enthält alle im konkreten Zusammenhang sinnvollen Aktionen. In allen Menüs werden Kommandos, die zur Zeit keinen Sinn ergeben oder nicht benutzt werden können, ausgegraut dargestellt.

Während länger andauernder Vorgänge blendet OTW-2 ein Statusfenster ein, das den Benutzer über den Vorgang informiert und den Arbeitsfortschritt darstellt. Beim Codegenerator und Scanner wird zum Beispiel das gerade bearbeitete Modul und die Zeile im Code angezeigt. Bei kürzeren Vorgängen wird eventuell nur die Cursor-Sanduhr aktiviert, um den Benutzer über den Programmstatus zu informieren.

b) Tool-Dokumentation

Als Dokumentation erhält der Benutzer neben dem Programm ein fast 500 Seiten umfassendes MS-Word Dokument, welches zugleich als Hilfesystem fungiert und aus dem Programm heraus aufgerufen werden kann. Dieses beinhaltet neben einer ausführlichen Bedienungsanleitung auch ein *Erste Schritte*-Kapitel, das überblicksweise ein mit der UML verbundenes Vorgehensmodell beschreibt. Weiters ist in diesem Dokument eine Installations- und Inbetriebnahmeanleitung enthalten.

Tool-Dokumentation, S. 48

1.	*+*
2.	*+*
3.	*0*
4.	*-*
5.	*+*
6.	*+*
7.	*-*
8.	*0*
9.	*0*
10.	*+*
11.	*+*
12.	*+*

Es existiert ein *Erste Schritte*-Handbuch in elektronischer Form, welches die Modellierung eines Projektes beschreibt. Hierbei wird mit vielen Screenshots die Arbeitsweise der OTW-2 demonstriert.

Mit der OTW-2 wird ein gedrucktes Benutzerhandbuch ausgeliefert, das die Bedienung der OTW-2 und Grundlagen der UML gleicherweise beschreibt. Es entspricht im wesentlichen dem Onlinehandbuch der OTW. Das Handbuch besitzt ein Stichwortverzeichnis.

Über letzte Änderungen und Neuigkeiten wird der Benutzer über eine *Readme*-Datei informiert. Außerdem existiert eine Datei, die Lösungsvorschläge zu häufigen Problemen anbietet. Mit OTW-2 werden einige fertig modellierte Beispiele ausgeliefert, die dem Benutzer als Lernhilfe dienen.

c) Usability

OTW-2 bietet dem Benutzer bei längeren Aktionen nicht die Möglichkeit, diese zu unterbrechen. Bei einem Test entstand der Eindruck, der Codegenerator und der Scanner ließen sich mittels *Escape*-Taste von ihrer Arbeit abbringen, da das Statusfenster, das über den Fortschritt informiert, augenblicklich verschwand, und auch die sonst übliche Benachrichtigung über den

erfolgreichen Abschluß der Aktion ausblieb. Genaueres Hinsehen offenbarte jedoch, daß die Aktion genauso durchgeführt wurde, wie dies ohne einen Druck auf die Escape-Taste erfolgt wäre.

Usability, S. 48

1.	0
2.	0
3.	0
4.	-
5.	+
6.	+
7.	0
8.	+
9.	+
10.	+
11.	0

Fehlermeldungen des Programmes sind bei OTW-2 selten, da während des Modellierens ständig die Konsistenz des Modells überwacht wird, und andererseits unsinnige Benutzerbefehle durch inaktive Menüeinträge verhindert werden. Eine nicht gefundene Datei zum Beispiel löst eine Fehlermeldung aus, die mit OK bestätigt werden muß, jedoch keine weiterführende Hilfe anbietet.

Wie schon erwähnt, sind die Kontextmenüs, die OTW-2 zur Verfügung stellt, sehr umfangreich. Sie bieten meist noch einige weitere Popup-Einträge und sind durch Separatoren übersichtlich gestaltet. Das Programm bietet eine horizontale Werkzeugleiste am oberen und eine vertikale am unteren Fensterrand, während die horizontale Leiste fix vorgegebene Buttons aufweist und dazu dient, gewünschte Ansichten zu öffnen bzw. in den Vordergrund zu bringen. Die vertikale Werkzeugleiste paßt sich dynamisch an das jeweils aktive Fenster an und stellt die wichtigsten Befehle als Buttons zur Verfügung.

Es sind nur drei Tastatur-Shortcuts im Programm implementiert, diese entsprechen voll dem Windows-Standard: *Entfernen* für löschen, *Control-A* für alles markieren und *F1* für das Benutzerhandbuch.

Das Hauptrahmenfenster ist mit einer Statuszeile am unteren Rand ausgestattet. Darin werden Kurzinformationen zu angewählten Menüpunkten und zu Schaltflächen der Werkzeugleisten angezeigt. Weiters wird im rechten Teil der Statuszeile ein Balkendiagramm angezeigt, das eine Übersichtlichkeitsbewertung des Diagramms darstellt. Diese soll einen Anhaltspunkt für den Modellierer liefern, wie das Diagramm durch den Leser erfaßt werden kann. Es gibt in diesem Diagramm folgende drei Ergebniskategorien:

- *trivial (gelb)* - das Diagramm enthält stark triviale Anteile (z.B. keine Kanten)
- *gut (grün)* - nicht trivial und überladen, Inhalt gut erfaßbar
- *überladen (rot)* - Diagramm enthält schwer erfaßbare Anteile (z.B. zu viele Elemente)

OTW-2 kann beliebig viele Ansichtsfenster gleichzeitig darstellen. Diese werden als Kindfenster innerhalb des Hauptrahmenfensters angelegt und sind in Position und Größe frei manipulierbar. Listenfelder werden vom Programm im Windows-Standard dargestellt. Wenn in Listenfeldern eine Mehrfachauswahl Sinn macht, ist sie auch verfügbar. Dialoge können in der OTW-2 stets auch rückwärts (abbrechen) verlassen werden.

OTW-2 ist nur in Deutsch erhältlich.

d) Hilfefunktionen

Die kontextsensitive Hilfe von OTW-2 wird über einen Menüeintrag oder *F1* aufgerufen. Sie ist ungewöhnlicherweise als MS-Word Datei realisiert, was ihren Gebrauchswert etwas einschränkt. Es sind Anker definiert, zu denen beim Aufruf in einem bestimmten Zusammenhang verzweigt wird, jedoch stehen Navigations- und Suchfunktionen innerhalb der Hilfedatei nicht so bequem zur Verfügung wie dies bei richtigen Hypertextformaten der Fall wäre. Außerdem ist die Dokumentation auf mehrere MS-Word Dokumente verteilt, was die Navigation innerhalb der Online-Hilfe weiter erschwert.

Hilfefunktionen, S. 49

1.	0
2.	0
3.	0
4.	+

Andererseits ist die Online-Dokumentation sehr ausführlich und umfangreich, und auch optisch gelungen. Sie enthält viele Screenshots aus dem Programm, die das Verständnis für die Erklärungen erleichtern. Der Haupt-

teil der Online-Hilfe ist ein 490 Seiten starkes Dokument, das fast wie ein Buch geschrieben ist und kaum Wünsche offenläßt.

OTW-2 bietet außer der Online-Dokumentation auch Tooltips an, das heißt, ein kleines Textfeld neben dem Cursor gibt dem Benutzer eine kurze Hilfestellung zu dem unter dem Cursor liegenden Steuerelement.

e) Graphikeditor / Darstellung

Graphikeditor / Darstellung, S. 49

1.	+
2.	-
3.	+
4.	+
5.	-
6.	+
7.	-
8.	-
9.	+
10.	+
11.	-
12.	-
13.	+
14.	+
15.	+

Da die OTW-2 keine Zustands- und Aktivitätsdiagramme unterstützt, ist der Grafikeditor auch nicht in der Lage, die Notationselemente dieser Diagramme darzustellen. Auch bei den anderen Diagrammarten fehlen einige in der UML spezifizierte Konstrukte (z.B. ternäre Assoziation, Qualifizierer, Shared Target Style). Die grafischen Elemente im Editor lassen sich über den Darstellungs-Dialog in gewissem Rahmen optisch verändern. So lassen sich Füllfarbe, Linienfarbe, Schriftart und -größe verändern. Die Strichstärke und die Linienart lassen sich nicht variieren. Außerdem kann bei der Klassendiagrammdarstellung neben der UML auch die Notation nach Brachman [Brac95] gewählt werden. Der Editor stellt keine allgemeinen Zeichenfunktionen bereit. Es ist dem Benutzer also unmöglich, UML-Konstrukte, die von OTW-2 nicht bereitgestellt werden, selbst zu erstellen.

Der Editor ist nicht in der Lage, Text oder Modellelemente zu rotieren. Im Programm ist keine automatische Layoutfunktion integriert, es besteht jedoch die Möglichkeit, grafische Elemente vom Editor auf einem Raster ausrichten zu lassen. Um die Optik des Diagramms weiter zu verbessern und Überschneidungen zu vermeiden, können in bestehende Kanten jederzeit neue Knickpunkte eingefügt werden.

Um die Ansicht zu skalieren, stehen dem Anwender zwei Zoom-Buttons in der vertikalen Werkzeugleiste zur Verfügung. Eine dient dazu, die aktive Ansicht um 25 Prozent zu vergrößern, die andere verkleinert sie um denselben Wert. Über das Fenster-Menü kann der Benutzer den Zoomfaktor stufenlos per Scrollbalken einstellen. OTW-2 bietet keine Funktion, um ein Diagramm von der Größe her in ein Fenster einzupassen. Ebenfalls auf der vertikalen Werkzeugleiste befindet sich ein Button, der das Programm dazu veranlaßt, alle geöffneten Diagrammansichten neu zu zeichnen.

Die Ausmaße der Klassensymbole im Grafikeditor können vom Benutzer nicht verändert werden. Sie passen ihre Größe dem Inhalt an und sind auch innerhalb eines Klassendiagramms nicht einheitlich. Ein Zeilenumbruch bei einer Textzeile innerhalb eines Klassensymbols ist nicht möglich.

Das Programm stellt die Diagramme am Bildschirm so dar, wie letztendlich auch der Ausdruck erfolgt. Um die Lage des zu druckenden Diagrammes auf der Druckseite zu überprüfen steht dem Benutzer eine Druckvorschau zur Verfügung.

f) Graphikeditor / Bedienung

Soll ein Notationselement in ein Diagramm eingefügt werden, muß der Benutzer zunächst die gewünschte Elementart über einen Button der vertikalen Werkzeugleiste oder das Kontextmenü auswählen. Durch diese Auswahl verändert sich der Mauszeiger und stellt jetzt symbolisch das einzufügende Element dar. Angelegt wird das Element durch einen Mausklick auf die Diagrammansicht, wodurch automatisch ein Dialog geöffnet wird, in dem ein vorhandenes Element selektiert, oder die Eigenschaften eines neuen Elementes spezifiziert werden können. Wird dieser Dialog vorwärts (mit *OK*) verlassen, stellt der Grafikeditor das angelegte Element an der durch den Mausklick vorgegebenen Stelle dar. Das Programm befindet sich weiterhin

im Einfügemodus für die zuvor gewählte Elementart bis der Benutzer über Werkzeugleiste oder Kontextmenü wieder das *verschieben/auswählen*-Werkzeug wählt.

Graphikeditor / Bedienung, S. 50

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	0
8.	0
9.	+
10.	+
11.	-
12.	0
13.	-
14.	+

Im Diagramm vorhandene Modellelemente können per Drag&Drop frei auf der unbegrenzten, vertikal und horizontal scrollbaren Zeichenfläche, positioniert werden. Um die Ausrichtung der Elemente auf der Zeichenfläche einheitlich zu gestalten, steht eine Zeichenraster-Funktion zur Verfügung. Dieses Raster ist jedoch von der Auflösung her nicht manipulierbar.

Das Programm erlaubt dem Benutzer, mehrere Modellelemente gleichzeitig zu selektieren. Dies geschieht entweder durch Ziehen eines Rahmen um die gewünschten Objekte oder durch ein Markieren mit der Maus bei gedrückter *Shift*- oder *Steuerung*-Taste. Auf diese Gruppe von Modellelementen können jetzt Operationen wie zum Beispiel Löschen oder Verschieben ausgeführt werden. Das Verschieben mehrerer Elemente zugleich gestaltet sich jedoch sehr kompliziert: Sind mehrere Elemente markiert und die Maustaste ist losgelassen, können diese Elemente nicht mehr verschoben werden, da bei einem erneuten Mausklick sofort die Mehrfachmarkierung aufgehoben und ein einzelnes Element selektiert wird. Mehrfachverschiebungen sind nur möglich, wenn die Markierung aller Elemente mit gedrückter Shift- oder Steuerung-Taste erfolgt, wobei diese auch während des Verschiebens gehalten werden muß.

OTW-2 besitzt eine Undo-Funktion, die einen Arbeitsschritt rückgängig macht. Eine Redo-Funktion wird nicht geboten.

Die Windowsfunktionen zum Datenaustausch über die Zwischenablage stehen im Grafikeditor nicht zur Verfügung. Notationselemente lassen sich einfach durch selektieren und drücken der Entfernen-Taste aus einem Diagramm löschen.

Insgesamt kann der Diagrammeditor der OTW-2 als leicht zu bedienen beschrieben werden, jedoch kann der Benutzer das Diagrammlayout nicht hundertprozentig selbst bestimmen. Bei allen Arten von Beziehungskanten können zwar zusätzliche Knickpunkte eingefügt, und diese nach belieben verschoben werden, die Kante hat ihren Ursprung jedoch immer im Zentrum des zugehörigen Klassensymbols. Besonders störend wirkt sich dieser Umstand aus, wenn mehrere Beziehungskanten auf der gleichen Seite auf das Klassensymbol treffen.

g) Performance

Performance, S. 51

1.	+
2.	0
3.	0
4.	+

In Sachen Performance gibt die OTW-2 keinen Grund zur Beanstandung. Sowohl die Darstellung der grafischen Ansicht, als auch der Codegenerator/ Scanner können in bezug auf die Geschwindigkeit überzeugen. Einzig das Laden eines neuen Repositorys nimmt mehr Zeit in Anspruch, als man aufgrund der Dateigröße vermuten würde, dabei muß jedoch berücksichtigt werden, daß das Programm initialisiert und die Ansichten aktualisiert werden müssen.

h) Adaptivität

Adaptivität, S. 51

1.	-
2.	-
3.	-
4.	-
5.	-
6.	-

Die Oberfläche von OTW-2 bietet dem Benutzer kaum Möglichkeiten, sie individuell anzupassen. Lediglich die Werkzeugleisten und die Statuszeile lassen sich auf Wunsch ein- und ausblenden.

Das Programm kennt keine Makro- oder Skriptsprache und wird auch nicht im Sourcecode ausgeliefert.

5.4 Paradigm Plus Enterprise 3.6

Das Softwareentwicklungswerkzeug Paradigm Plus der Firma Platinum Technology ist ein einfach zu bedienendes, aber sehr leistungsfähiges Programm. Die verwendeten Konstrukte halten sich eng an den UML-Standard.

Als Repository kommt die objektorientierte Datenbank Objectstore zur Verwendung, die ein ausreichendes Maß an Funktionalität zur Verfügung stellt. Die Generierung von Quellcode sowie das Reverse-Engineering werden korrekt und effizient ausgeführt. Der mögliche Einsatzbereich von Paradigm Plus reicht von kleineren bis zu komplexen, verteilten Projekten mit einer großen Anzahl von Benutzern.

5.4.1 Methodenbasierte Kriterien

a) Modellierung der Objektstruktur - Objektstatik

Modellierung der Objektstruktur - Objektstatik, S. 38

1.	+
2.	+
3.	0
4.	+
5.	+
6.	0
7.	+

Beim Anlegen neuer Elemente wird nicht automatisch auf Namenskonflikte überprüft, erst beim Ändern des Namens (siehe *Konsistenzprüfung/Integrität*, S. 94). Die Reihenfolge der Komponenten kann grundsätzlich selbst bestimmt werden, sie richtet sich jedoch nur nach der Reihenfolge der Eingabe und kann nachträglich nicht mehr verändert werden. Parameter einer Methode können prinzipiell spezifiziert werden, jedoch steht hierfür nur ein einziges Feld zur Verfügung, sodaß auch keine weiteren Erläuterungen (wie Eingangs-, Ausgangsparameter) angegeben werden können.

b) Modellierung der Systemstruktur - Systemstatik

Modellierung der Systemstruktur - Systemstatik, S. 39

1.	+
2.	-
3.	-
4.	0

Einmal erfaßte Beziehungen können nicht nachträglich verändert werden. Es besteht nur die Möglichkeit, die Beziehung zu löschen und neu einzutragen. Die Einteilung von Klassen erfolgt nur mittels Packages (*Components*). OCL (*Object Constraint Language*)-Annotationen müssen vom Benutzer eingegeben werden und werden vom Werkzeug nicht weiter interpretiert.

c) Dynamisches Objektverhalten - Objektdynamik

Dynamisches Objektverhalten - Objektdynamik, S. 39

1.	0
2.	0
3.	0
4.	+
5.	+
6.	-

Bei der Erstellung von Zustandsdiagrammen sind diese nicht unbedingt zwingend mit einer Klasse verbunden. Eine Möglichkeit besteht darin, das Zustandsdiagramm als sog. *Child* einer Klasse zu modellieren (Auswahl von *New Child...* im Kontextmenü der Klasse). Danach kann auch bequem zum entsprechenden Diagramm navigiert werden (siehe *Navigation im Systemmodell*, S. 94). Jedoch ist diese Funktion nicht eingeschränkt, so kann jeder beliebige Diagrammtyp als *Child* eines bestimmten Elements angegeben werden.

Weiters kann auch zuerst die Klasse im State-Diagram eingezeichnet und dann die Zustände innerhalb der Klasse plaziert werden, dies ist jedoch eher fehlerträchtig und unangenehm.

Die Angabe von Bedingungen für Zustandsübergänge ist möglich, jedoch stehen nur Textfelder zur Verfügung, sodaß nicht direkt auf Attribute der Objekte Bezug genommen werden kann. Eine kurze Beschreibung von Aktivitäten u.a. ist möglich (Textfeld).

Das Erstellen von ODER-generalisierten Zuständen ist problemlos möglich, jedoch werden UND-verfeinerte Zustände nicht ganz UML-konform dargestellt (Fehlen der Trennlinie im Superzustand; die einzelnen Subzustände

müssen im Eigenschaftendialog als *concurrent* spezifiziert werden und werden danach mit einem strichlierten Rahmen angezeigt).

d) Dynamisches Systemverhalten - Systemdynamik

Dynamisches Systemverhalten - Systemdynamik, S. 39

1.	0
2.	0

Beim Erstellen von Sequenzdiagrammen können in einem Dialogfenster vorhandene Methoden eines Objekts ausgewählt werden, auf Attribute o.ä. kann jedoch nicht Bezug genommen werden.

Wird im Dialog der Bezeichner einer neuen Operation eingetragen, so speichert Paradigm Plus diese in der Datenbasis für die entsprechende Klasse. Die Operation kann danach über die Funktion *Populate* im Klassendiagramm automatisch hinzugefügt werden.

e) Zusammenhang zwischen den Modellen

Zusammenhang zwischen den Modellen, S. 39

1.	0
2.	-
3.	+
4.	-

Der Zusammenhang zwischen einer Klasse (Objekt) und dem dazugehörigen Zustandsdiagramm muß vom Benutzer explizit modelliert werden (siehe *Dynamisches Objektverhalten - Objektdynamik*, S. 92). Das dynamische Verhalten einer Klasse wird auch nicht an deren Unterklassen vererbt.

f) Abstraktionsebenen / Sichten

Abstraktionsebenen / Sichten, S. 40

1.	+
2.	+
3.	-
4.	-
5.	0

Über die Funktion des *Diagram-Filter* kann in einem Dialogfenster detailliert ausgewählt werden, welche Elemente eines Diagramms ein- bzw. ausgeblendet werden sollen. Dies beschränkt sich nicht nur auf bestimmte Elemente, wie Attribute oder Operationen - für prinzipiell alle vorkommenden Teile eines Diagramms kann eine entsprechende Einstellung der Anzeige vorgenommen werden. Außerdem ist es möglich, die auszublendenden Elemente nur in grauer Farbe darzustellen.

Der Filter bezieht sich auf ein bestimmtes Diagramm, es kann jedoch auf Wunsch automatisch eine Kopie des aktuellen Diagramms erstellt werden, welche die ausgewählten Ansichtseinstellungen enthält.

g) Vorgangsreihenfolge bei der Entwicklung

Vorgangsreihenfolge bei der Entwicklung, S. 40

1.	+
2.	+
3.	+
4.	-

Da die Verwaltung des Repositorys unabhängig von den Diagrammsichten besteht, wird der Benutzer nicht zu einer bestimmten Vorgangsreihenfolge gezwungen. Die Daten können über den Repository-Browser eingetragen, verändert, und später in Diagrammen verwendet werden. Das Repository wird - je nach Einstellung - automatisch beim Anlegen neuer Elemente in einem Diagramm oder erst nach Aufforderung aktualisiert. Somit ist es z.B. möglich, Entwürfe zu zeichnen, ohne daß die Daten sofort gespeichert werden.

h) Unterstützung bei der Methodenanwendung

Unterstützung bei der Methodenanwendung, S. 40

1.	+
2.	0
3.	+
4.	+
5.	-
6.	+

Es steht immer ein Diagramm (*UML-Notation*) mit den wichtigsten Notationsformen zu Verfügung.

Zusammen mit dem Tool werden mehrere Dokumente im PDF-Format mitgeliefert, die auf Wunsch auch in Buchform erhältlich sind. Es sind dies u.a. ein Methoden-Manual, das sämtliche verwendeten Methoden und deren Notation ausreichend beschreibt, weiters ein *User's Guide*, der durch die Benutzung des Tools führt und ein *Getting Started Guide*, der anhand eines durchgehenden Beispiels einen Einblick in die Diagrammerstellung gibt.

Eine automatische Warnfunktion für Konsistenzverletzungen ist nicht vorgesehen, auch erfolgt keine weiterführende Erklärung der Ergebnisse der Konsistenzprüfung.

5.4.2 Funktionale Kriterien

a) Datenhaltung

Alle Daten in Paradigm Plus werden über ein zentrales Repository verwaltet und stehen im echten Multi-User Betrieb allen Benutzern zur Verfügung. Dieses bildet den Kernstück des Tools und ist als objektorientierte Datenbank ausgeführt. Genutzt wird hierfür die Database Engine ObjectStore der Firma Object Design Inc.

Für jedes Projekt steht ein eigenes Repository zur Verfügung, das seinerseits jedoch andere referenzieren kann. Alle Objekte können im Repository gewartet werden, es ist nicht zwingend notwendig, daß Objekte des Repository auch in einem Diagramm vorkommen. So ist auch sichergestellt, daß Komponenten über mehrere Projekte hinweg wiederverwendet werden können, ohne mehrfach abgelegt werden zu müssen. Diagramme, die vom Benutzer erstellt werden, stellen hiermit im Prinzip verschiedene Ansichten des Repository dar. Wird ein wiederverwendetes Element in einem bestimmten Projekt geändert, so wirkt sich diese Änderung automatisch auf alle anderen Projekte aus.

Datenhaltung, S. 41

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	+
8.	+

b) Browsing-Eigenschaften

Die Suchfunktion nach bestimmten Begriffen steht im Diagrammeditor bzw. Ausgabefenster zur Verfügung. Diese unterstützt kein genaues Pattern-Matching, jedoch besteht die Möglichkeit, im *Object-Browser* (Repository-Fenster) bestimmte Elemente z.B. gemäß ihrem Anfangsbuchstaben zu suchen und ihre Eigenschaften zu verändern. Eine Ersetzen-Funktion ist ebenfalls verfügbar. Klassenhierarchien werden weitgehend übersichtlich dargestellt.

Browsing-Eigenschaften, S. 41

1.	+
2.	-
3.	+
4.	-
5.	-
6.	*0*
7.	+
8.	+
9.	-
10.	+
11.	-

c) Navigation im Systemmodell

Da Paradigm Plus mit einem Multiple Document Interface ausgestattet ist, können verschiedene Diagramme geöffnet, und schnell zwischen ihnen gewechselt werden. Es ist auch möglich, Diagramme bestimmten Modellierungselementen zuzuordnen, z.B. kann ein Zustandsdiagramm als Child Diagram einer Klasse zugeordnet werden. Über den Kontextmenüeintrag der Mutterklasse *To Child* im Klassendiagramm wird dann das zugeordnete Zustandsdiagramm geöffnet.

Sämtliche grafische Designelemente können per Mausklick angewählt werden und stellen in einem Properties-Dialog weiterführende Auskünfte bereit.

Navigation im Systemmodell, S. 42

1.	+
2.	+
3.	-
4.	+

d) Konsistenzprüfung/Integrität

Das Tool prüft nicht schon bei der Eingabe, ob diese sinnvoll ist. Auch leere Modellierungselemente sind möglich, sie werden vom Programm ignoriert.

Es können mehrere Elemente mit gleichem Namen angelegt werden. Diese werden nur einmal in das Repository aufgenommen, was zu merkwürdigen Situationen führen kann. Legt man zum Beispiel zwei Klassen mit gleichem Namen an, die eine mit einem Attribut A, die andere mit einem Attribut B, wird im Repository eine Klasse dieses Namens geführt, die beide Attribute enthält. Will man jedoch den Namen einer Klasse in den einer bereits bestehenden Klasse ändern, erfolgt eine Fehlermeldung.

Konsistenzprüfung / Integrität, S. 42

1.	-
2.	-
3.	+
4.	-
5.	-
6.	-
7.	+
8.	0
9.	+

Rekursive Vererbungshierarchien werden zugelassen und diese werden auch im Repository eingetragen und von der Konsistenzprüfung akzeptiert. Soll ein Designelement aus einem Diagramm gelöscht werden, wird zusätzlich abgefragt, ob das Element auch aus dem Repository entfernt werden soll.

Paradigm Plus verfügt über eine umfangreiche Konsistenzprüfung zwischen den statischen und den dynamischen Modellen, deren Ergebnis im Consistency Check Report dargestellt wird. Art und Umfang der Prüfungen hängen von der gewählten Methode und der Generierungssprache (C++, o.ä.) ab. Diese Reports werden mit Hilfe von Scripts erzeugt, die vom Benutzer an die entsprechenden Anforderungen angepaßt werden können.

e) Report-Ausgabe

Report-Ausgabe, S. 43

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	-
8.	+
9.	+
10.	0

Es kann aus Paradigm Plus heraus eine automatische Dokumentation der verwendeten Elemente generiert werden. Diese steht wahlweise in zwei verschiedenen Formen zur Verfügung. Die einfachere Möglichkeit ist eine Dokumentation in reinem Textformat. Diese wird über Scripts in Umfang und Inhalt bestimmt und kann so leicht den Erfordernissen angepaßt werden. Die zweite Art stellt der sogenannte *Webexport* in das HTML-Format dar, wobei Diagramme als JPG-Dateien oder als Applets zusätzlich eingebunden werden können. So können Ergebnisse von Projekten einfach im Internet oder im firmeneigenen Intranet anderen Mitarbeitern zur Verfügung gestellt werden, die dann auch die Möglichkeit haben, bestimmte Komponenten per Download in ihr eigenes Projekt einzubinden.

Über das ebenfalls von Platinum angebotene Zusatzwerkzeug ParaDoc können diese Arten der Dokumentation um verschiedenste zusätzliche Funktionen erweitert werden. ParaDoc ist ein Dokumentationsgenerator, der auf Basis von vordefinierten und änderbaren Vorlagen Dokumentationen im Rich Text Format (RTF) und als Windows Metafile (WMF) generieren kann, die anschließend von jedem Textverarbeitungsprogramm gelesen und verarbeitet werden können. ParaDoc nimmt hierzu nicht nur Bezug auf die Daten des aktuellen Repositorys, sondern bindet auch Informationen der im Projekt enthaltenen Diagramme und etwaiger externer Dokumente ein.

f) Datenbankschema-Ausgabe

Datenbankschema-Ausgabe, S. 43

1.	+
2.	+
3.	-
4.	+
5.	-
6.	+

Die Anbindung an die relationale Datenbankwelt ist in Paradigm Plus sehr benutzerfreundlich und effizient gestaltet. Ähnlich wie bei den Reports stehen grundsätzlich zwei Möglichkeiten zur Verfügung. Zum einen können direkt Entity-Relationship-Diagramme erstellt und bearbeitet werden (ähnlich wie UML-, OMT-, und andere Diagramme), mit allen zugehörigen Informationen, wie Spalten, Relationen, Tabellen etc. Durch einen sogenannten *Mapping*-Mechanismus können auch die im Objektmodell enthaltenen Informationen auf solche ER-Diagramme direkt abgebildet werden. Diese Vorgehensweise erstreckt sich auch auf das physikalische Modell. Auf der anderen Seite steht die Anbindung an das Datenmodellierungssystem ERwin (ebenfalls von Platinum) zur Verfügung. Beide Tools können bidirektional Infor-

mationen austauschen und so kann etwa das komplette Objektmodell in ERwin übernommen werden, um so in einfacher Weise Datenbanken zu designen und zu implementieren.

g) Diagramm-Ausgabe

Das Tool bietet eine Windows-konforme Druckschnittstelle. Die Druckerausgabe arbeitet nach dem *What You See Is What You Get*-Prinzip. Auf Wunsch kann in Diagrammen direkt der Druckbereich angezeigt werden.

Möglichkeiten, den Ausdruck zu konfigurieren bestehen im Papierformat, Hoch-/Querformat, der Skalierung, Anpassung an die Seite, Kopf- und Fußzeilen und anderem.

Diagramm-Ausgabe, S. 44

1.	+
2.	0
3.	-
4.	+
5.	0
6.	+

h) Offenheit - Import/Export

Paradigm Plus verfügt über eine große Anzahl von Schittstellen, sowohl zu Produkten von Platinum selbst (z.B. ERwin oder ParaDoc), als auch zu Produkten von Drittherstellern in den Bereichen Projekt-Management, Datenbank-Management, Konfigurations- und Versionsmanagement, u.a. Die Schnittstellen selbst können in drei Kategorien eingeteilt werden. Erstens in die reinen Export- und Importschnittstellen (*CDIF* - Case Data Interchange Format, ein spezielles ASCII-Format für Modellbeschreibungen, das auch andere CASE-Tools nutzen, sowie eine allgemeine ASCII-Schnittstelle, die von vielen Drittwerkzeugen, z.B. MS-Excel und Project gelesen werden kann), zweitens die Kommandoschnittstelle, über die Drittprogramme von Paradigm Plus aus angesteuert werden könne, ohne die Oberfläche verlassen zu müssen. An dritter Stelle folgen die API-Schnittstellen, die zu den gängigsten Werkzeugen bestehen, wie etwa Symantec Visual Café, Microsoft Developer Studio, oder zu ObjectStore Active/X SDK für den externen Zugriff auf das Repository.

Der Datenaustausch von Texten oder Bilddateien via Cut&Paste über die Zwischenablage wird von Paradigm Plus ebenfalls anstandslos unterstützt. Eingebundener Text kann als String direkt im Tool editiert werden. Für Bilddateien (BMP, JPG etc.) steht eine derartige Bearbeitungmöglichkeit nicht zur Verfügung, auch die Einbettung über OLE wird nicht zur Verfügung gestellt, sodaß ein bereits eingefügtes Objekt nicht durch Auswahl (Doppelklick) im zugeordneten Programm (z.B. Paint) verändert werden kann.

Offenheit - Import / Export, S. 44

1.	+
2.	+
3.	-
4.	+
5.	+
6.	+
7.	+
8.	0

i) Code-Generierung

Aus den im aktuellen Projekt enthaltenen Daten generiert Paradigm Plus auf Wunsch die Coderahmen für die Implementierung. Das Tool wird mit einer großen Zahl von Codegeneratoren geliefert, so etwa (Visual) C++, Java, Smalltalk, Visual Basic, Delphi, relationale und objektorientierte Datenbanken, u.a. Diese können auch ohne großen Aufwand über Scripts an die jeweiligen Anforderungen angepaßt werden. Für jede Klasse werden eigene Source Files angelegt, Attribute werden mit ihren Typen, Methoden mit Parametern und Rückgabetypen implementiert.

In diesem Zusammenhang ist besonders der effiziente Mechanismus der *inkrementellen Codegenerierung* hervorzuheben, wobei vom Entwickler selbst hinzugefügter Code in eigenen Zwischendateien ausgelagert und bei erneuter Erzeugung vom Generator dem neuen Gerüst wieder hinzugefügt wird. Bei allfälligen Änderungen der Methodensignaturen wird die alte Signatur als Kommentar beibehalten. Paradigm Plus kommt hierbei auch ohne Code-Markierungen aus.

Code-Generierung, S. 45

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	+
8.	0
9.	0
10.	+
11.	+

j) Reverse-Engineering

Reverse-Engineering, S. 45

1.	+
2.	+
3.	+

Um die Möglichkeiten des sog. Round-Trip-Engineering zu gewährleisten, kann Paradigm Plus bestehenden Programmcode einlesen, auch wenn er nicht mit dem Tool erstellt wurde und importiert diesen in das aktuelle Repository. Diese Daten kann man nun in verschiedenen Diagrammen darstellen (über die Funktion *Populate*, bei der die gewünschten Elemente der Datenbasis im Diagramm dargestellt werden). Das resultierende Modell entspricht sehr genau den, im Quellcode enthaltenen Informationen. Das Werkzeug erkennt auch Beziehungen zwischen Klassen (Assoziationen, Generalisationen etc.) ohne Probleme und setzt diese exakt im entsprechenden Klassendiagramm um.

Code kann auch direkt in Paradigm Plus editiert werden, da die Möglichkeit besteht, für jede Klasse eine Quelldatei anzugeben. Diese kann vom Eigenschaftendialog der Klasse aus eingesehen und verändert werden. Der so erstellte Code wird vom Codegenerator nicht entfernt (siehe *Code-Generierung*, S. 96).

k) Test / Verifikation

Test / Verifikation, S. 45

1.	+
2.	-

Zur Generierung von Testfällen bietet das Tool die Möglichkeit, eine Beschreibung dieser im Repository zu verwalten, wobei auch eine Verknüpfung mit den Modelldaten möglich ist. Über Reports können die Aktivitäten der Tests dokumentiert werden.

l) Maßzahlen und Statistiken

Maßzahlen und Statistiken, S. 46

1.	+
2.	+
3.	*0*
4.	+

Paradigm Plus bietet eine Reihe von vorgefertigten Reports, die unter anderem ausführliche Statistiken und Metriken über ein Projekt ausgeben. So stehen standardmäßig verschiedene Scripts zur Verfügung:

- ❑ Analyse-Metriken
  (z.B. Kopplung von Klassen, Tiefe der Vererbungshierarchie etc.)
- ❑ Design-Metriken
  (z.B. Anzahl öffentlicher Methoden in Klassen, Anzahl von Instanzvariablen in Klassen etc.)

Mit dem Zusatztool Platinum ADvisor lassen sich die Daten des Repositorys beliebig auswerten und etwa für das Projektmanagement verwenden.

m) Multi-User - Eigenschaften

Multi-User - Eigenschaften, S. 46

1.	+
2.	+
3.	*0*
4.	+
5.	*0*
6.	-
7.	-

Paradigm Plus ist sowohl auf Windows- als auch auf UNIX-Plattformen lauffähig und kann im Multi-User Betrieb sowohl in homogenen als auch in heterogenen Umgebungen betrieben werden, wobei PC- und UNIX-Clients auf das gleiche Repository zugreifen können.

Die Mehrbenutzerkontrolle an sich wird durch das OODBMS ObjectStore, welches das gesamte Repository organisiert und verwaltet, abgewickelt. Dadurch stehen sämtliche Security- und Recovery-Möglichkeiten dieses DBMS zu Verfügung.

Funktionen zum interaktiven Dialog zwischen den Benutzern (Nachrichten, *Talk*- Funktion) sind nicht vorhanden.

n) Security

Security, S. 46

1.	+
2.	0
3.	-
4.	0
5.	+
6.	0
7.	+
8.	0
9.	0
10.	+
11.	0

Im Multi-User-Betrieb ist jedes Repository benutzbar. So kommen verschiedene Sperr-Mechanismen zur Anwendung, die auf verschiedenen Ebenen wirken können. Der Benutzer kann somit einzelne Elemente oder ganze Diagramme mittels einer *Lock*-Funktion sperren, wahlweise mit Schreib-, oder Nur-Lese-Zugriff. Eine Klasse kann z.B. von einem Entwickler bearbeitet werden und steht dennoch anderen zur Ansicht zur Verfügung. Außerdem müssen sich auf Wunsch alle Benutzer vor der Verwendung eines bestimmten Repositorys mit Usernamen und Passwort ausweisen. Auch Änderungen eines Users können während der Arbeit protokolliert werden.

Die Fehlerbehandlung ist ausreichend, es werden Fehlermeldungen in Form von Messageboxen ausgegeben. Programmabstürze infolge von Bedienungsfehlern traten während des Arbeitens nicht auf.

Paradigm Plus bietet die Möglichkeit der Aktivierung einer Sicherheitsabfrage vor dem Löschen, eine Undo-Funktion ist ebenfalls vorhanden.

o) Multi-Projekt-Eigenschaften/ Versionsmanagement

Multi-Projekt-Eigenschaften / Versionsmanagement, S. 47

1.	+
2.	+
3.	+
4.	+
5.	0

Paradigm Plus unterstützt in hohem Maße die parallele Abwicklung verschiedener Projekte innerhalb einer Arbeitsumgebung und die Aufteilung sowie Wiederverwendung vom Komponenten. Bereits erstellte Konstrukte können einfach in ein anderes Projekt importiert werden, auch Referenzen auf diese können eingetragen werden.

Das Tool kann mehrere Versionen eines Projekts und seiner Komponenten verwalten, wobei ein Projekt während seines Entwicklungs-Lebenszyklus anhand von Meilensteinen (*baselines*) und Komponenten durch verschiedene Versionsnummern unterschieden werden. Dies erlaubt auch ein automatisiertes *Configuration Management* (CM), indem ein Projekt in verschiedenen lauffähigen Konfigurationen (Versionen) verschiedene Komponenten beinhalten kann und so ein genaues Austesten ermöglicht wird. Hierbei ist auch eine Anbindung an verschiedene CM-Tools (CCC/Harvest, PVCS, MS-Visual SourceSafe) integriert.

5.4.3 Ergonomie

a) Benutzeroberfläche

Benutzeroberfläche, S. 47

1.	0
2.	+
3.	+
4.	+
5.	+
6.	0
7.	+

Die getestete Windows-Version von Paradigm Plus besitzt das Look-and Feel eines echten Windows-Programms, obwohl es von einem UNIX-Programm portiert wurde. Optisch entspricht die Oberfläche jedoch einer älteren Anwendung für Windows 3.x.

Die mitunter umfangreichen Dialoge und Eingabemasken sind einigermaßen übersichtlich gestaltet, die Reihenfolge in der die einzelnen Controls mit dem Tabulator aktiviert werden ist jedoch nicht immer intuitiv.

Längere Aktionen werden mit einer Sanduhr angezeigt, eine Fortschrittsanzeige fehlt jedoch.

b) Tool-Dokumentation

Die Dokumentation zu Paradigm Plus liegt in Form einiger vorbildlich gestalteter Files im Portable Document Format (PDF) auf der Installations-CD bei. Optional kann sie jedoch auch in gedruckter Form bestellt werden. Folgende

Handbücher werden zur Verfügung gestellt: Starthandbuch, Benutzerhandbuch, Script-Language Referenz, Methodenhandbuch und Installationshandbuch.

Versierten Benutzern wird die Lektüre der Dokumentation durch Zusammenfassungen zu jedem Kapitel und einen Referenzteil erleichtert. Anfänger bekommen in einem *Getting started* Kapitel einen Überblick über das Tool.

Die PDF-Files fungieren in nicht gedruckter Form vor allem als Online-Referenz. Sie besitzen allesamt ein Inhaltsverzeichnis, mit dem man zielsicher durch die Anleitung navigieren kann. Weiters ist eine bequeme Suche nach benötigten Informationen dadurch gewährleistet, daß im Hilfe-Menü die Inhalte dieser Handbücher nach Themen geordnet im Hyperlink-Format bereitstehen. Mit Paradigm Plus werden außerdem eine Reihe teils sehr umfangreicher Beispielprojekte geliefert.

Tool-Dokumentation, S. 48

1.	+
2.	+
3.	-
4.	0
5.	+
6.	+
7.	0
8.	+
9.	+
10.	+
11.	+
12.	+

c) Usability

Das Programm bietet im Fehlerfall keine weiterführende Hilfe an, der Fehler wird in einer Messagebox textuell erläutert.

Es werden Tastatur-Shortcuts von gängigen Befehlen unterstützt, diese beschränken sich jedoch auf das Edit-Menü und das Aufrufen verschiedener Ansichten.

Das Programm bietet ein Kontextmenü mit allen wichtigen Befehlen, welches man leicht über die rechte Maustaste erreichen kann. Die Dialoge sind ganz im Windows Standard über *OK* und *Abbruch* zu verlassen. Toolbars sind zwar vorhanden, doch lassen sich diese nicht anpassen. Eine Statuszeile ist zwar vorhanden, doch leider wird dort nur angezeigt, ob das System bereit oder beschäftigt ist (leider ohne Fortschrittsanzeige). Das Tool ist mit einem Multiple Document Interface ausgestattet.

Paradigm Plus ist leider nur in englischer Sprache erhältlich.

Usability, S. 48

1.	0
2.	0
3.	+
4.	0
5.	+
6.	0
7.	0
8.	0
9.	+
10.	0
11.	-

d) Hilfefunktionen

Das Hilfesystem von Paradigm Plus basiert auf einem übersichtlich gegliederten Windows Helpfile. Kontextsensitive Hilfe und Hot-Help werden nicht geboten.

Hilfefunktionen, S. 49

1.	0
2.	0
3.	+
4.	-

e) Graphikeditor / Darstellung

Im Großen und Ganzen können alle UML-Notationselemente dargestellt werden, einige wenige Ausnahmen werden etwas abgewandelt gezeichnet.

Der Editor hält eine breite Palette von Zeichnungselementen bereit, mit denen sich freie, nicht standardisierte Objekte zeichnen lassen. Hierzu zählen zum Beispiel Zeichenfunktionen für die gängigsten geometrischen Figuren und ein sehr großes Sortiment an Pfeilspitzen.

Das Tool bietet viel gestalterische Freiheit. Abgesehen von der Schriftgröße sind auch noch Font und Farbe anpaßbar, verschiedene Strichstärken für die Zeichnungselemente und Verbindungslinien lassen sich bequem einstellen. Es stehen die 16 Systemfarben von Windows zur Verfügung um damit Linien, ganze Objekte oder Text einzufärben. Zum Rotieren von Text oder Zeichnungselementen wird leider keine Funktion bereitgestellt, ebensowenig wie eine Refresh-Funktion.

Es gibt eine Reihe von Alignment-Funktionen, die alle sehr nützlich sind, die automatische Layoutoptimierung ist jedoch, wie bei anderen Tools auch, ein wenig mit Vorsicht zu genießen. Die nachträgliche Manipulation von Relationen ist relativ umständlich, oft hilft nur ein Neuanlegen der Beziehung.

Graphikeditor / Darstellung, S. 49

1.	0
2.	+
3.	+
4.	+
5.	+
6.	+
7.	-
8.	0
9.	+
10.	+
11.	0
12.	-
13.	+
14.	+
15.	+

Die Ansichtsfenster lassen sich stufenlos zoomen. Das Layout eines Diagramms wird gespeichert.

Der Ausdruck solle der Ansicht im Diagrammfenster entsprechen, dies konnte jedoch mit unserer Testversion des Programms nicht einwandfrei getestet werden.

f) Graphikeditor / Bedienung

Graphikeditor / Bedienung, S. 50

Die Notationselemente werden aus einem vertikalen Toolbar gewählt und einfach auf die Zeichenfläche geklickt. Dort können sie per Drag & Drop verschoben werden. Grid Alignment steht auch zur Verfügung, zusätzlich kann man mehrere Zeichnungselemente wählen und mehrere andere Alignmentfunktionen auf sie anwenden.

Es gibt eine Undo, jedoch keine Redo-Funktion. Wir mußten jedoch feststellen, daß die Undo-Funktion in manchen Fällen nicht verfügbar ist (z.B. Layoutoptimierung).

Vorbildlich ist das Edit Menü, das die Standard-Bearbeitungsfunktionen anbietet, die auch mit den Tastatur-Shortcuts nach dem Windows-Standard versehen sind. Löschen beliebiger Elemente durch Markieren und Betätigen der *Entf*-Taste ist möglich. Alles in allem kann man den Grafikeditor als intuitiv betrachten.

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	0
8.	+
9.	0
10.	+
11.	-
12.	+
13.	+
14.	+

g) Performance

Die Geschwindigkeit des Programms liegt im Bereich etwas komplexerer Anwender-Software.

Performance, S. 51

Der Bildaufbau erfolgt recht zügig, wobei das Laden und Speichern der Daten natürlich eine gewisse Zeit in Anspruch nimmt. Im Allgemeinen ist jedoch eine flüssige Arbeit möglich. Auch die Codegenerierung bzw. das Wiedereinlesen von Code wird im Vergleich mit anderen Werkzeugen relativ schnell ausgeführt.

1.	+
2.	+
3.	+
4.	+

h) Adaptivität

Adaptivität, S. 51

Paradigm Plus bietet weitgehende Anpassungsmöglichkeiten an die Erfordernisse der Arbeit. Es steht eine auf der Syntax von Visual-Basic basierende Scriptsprache zur Verfügung, die auf sämtliche Daten des Repository zurückgreifen kann und so an vielen Stellen die Funktionalität erweitert. Bestimmte Scripts können auch verschiedenen Menüpunkten oder Benutzerinteraktionen zugeordnet und somit automatisch aufgerufen werden (z.B. Protokollierung, Konsistenzprüfungen, Start eines anderen Programms, spezielle Import-/Exportfilter etc.) Weiters können in den Diagrammeditoren verwendbare PopUp-Menüs erstellt und konfiguriert werden. So kann schnell auf häufig benötigte Funktionen zugegriffen werden.

Der Quellcode des Programms wird nicht mitgeliefert.

1.	0
2.	0
3.	+
4.	0
5.	+
6.	-

5.5 Rational Rose 98 Enterprise SR-1

Das Programm Rational Rose 98 der Firma Rational Software Corp. ist das mit Abstand verbreitetste Produkt zur Realisierung von Softwareprojekten mittels der UML. Dieser Vorsprung resultiert unter anderem daraus, daß die "Väter" der UML, Grady Booch, Ivar Jacobson und James Rumbaugh bei Rational arbeiten. Dies ist auch mit ein Grund dafür, daß Rational Rose die vorgegebenen Konzepte (Diagrammtypen, Darstellung der Elemente etc.) nahezu hundertprozentig dem Standard entsprechend umsetzt, wenngleich auch nicht immer gemäß der neuesten Version des Standards.

Im Jahr 1999 wurde Rational Rose von der IDC (International Data Corporation) zum vierten Mal hintereinander zum erfolgreichsten objektorientierten Analyse- und Designtool erklärt (gemessen am erwirtschafteten Gewinn). Der Marktanteil (ebenfalls gemessen am Gewinn) von Rational Rose übertraf 1998 die kumulierten Anteile seiner vier nächsten Konkurrenten.

5.5.1 Methodenbasierte Kriterien

a) Modellierung der Objektstruktur - Objektstatik

Modellierung der Objektstruktur - Objektstatik, S. 38

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	+

Das Programm speichert Informationen des Modells in einem Data Dictionary. In einem Browserfenster, das an die Hauptansicht gedockt ist, kann der Inhalt dieses Repositorys begutachtet und manipuliert werden. Notationselemente können mittels Browser direkt im Repository angelegt und manipuliert werden. Hieraus folgt, daß Elemente im Repository existieren können, die in keinem Diagramm vorkommen.

Bereits existierende Elemente können jederzeit manipuliert werden, Rational Rose prüft in einem solchen Fall, ob die Veränderungen mit den übrigen Modellinformationen verträglich sind (z.B. Namenskonflikt).

Attribute haben einen bestimmten Typ, einen Namen und einen Initialwert. Zusätzlich kann ihre Sichtbarkeit (*public*, *protected*, *private*), ihre Lebensdauer (*static*, *dynamic*) und einige andere Parameter festgelegt werden. Operationen besitzen neben Namen und Rückgabetyp und Sichtbarkeit noch eine Parameterliste. Weiters können Vor- bzw. Nachbedingungen angegeben werden, die erfüllt werden müssen.

b) Modellierung der Systemstruktur - Systemstatik

Modellierung der Systemstruktur - Systemstatik, S. 39

1.	+
2.	+
3.	-
4.	-

Rational Rose läßt die Modellierung von rekursiven Aggregations-, Abhängigkeits- und Assoziationsbeziehungen zu. Einmal erfaßte Beziehungen können zu einem beliebigen späteren Zeitpunkt manipuliert werden. Das Programm prüft in einem solchen Fall, ob die Veränderungen zulässig sind und verhindert gegebenenfalls ungültige Manipulationen. Es ist sogar möglich, nachträglich die beteiligten Parteien einer Beziehung zu verändern. Das bedeutet, der Benutzer kann per Drag & Drop die Beziehungskante von einem Notationselement lösen und sie mit einem anderen verbinden.

Klassen lassen sich abgesehen von der Zuweisung zu Packages nicht in Gruppen zusammenfassen. OCL-Notationen werden nicht unterstützt.

c) Dynamisches Objektverhalten - Objektdynamik

Für jede Klasse kann ein Zustandsdiagramm angelegt werden. Dieses wird eindeutig mit der entsprechenden Klasse verknüpft. Der Benutzer gelangt zu

dem Zustandsdiagramm einer Klasse über das Menü/Kontextmenü, wenn er im Klassendiagramm eine Klasse auswählt, beziehungsweise über den Browser, in dem diese als Bestandteil einer Klasse dargestellt werden.

Rational Rose erlaubt die Modellierung von normalen Zuständen, Start- und Endzuständen sowie einfachen und rekursiven Zustandsübergängen. Weiters können geschachtelte Zustände erzeugt werden. Auch History States und Deep History States können modelliert werden.

An Zustandsübergänge können Bedingungen geknüpft werden, die erfüllt sein müssen, damit der Übergang stattfindet. In den Bedingungen kann Bezug auf Attribute der zugrundeliegenden Klasse genommen werden. Von einem Zustandsübergang, aber auch von Zuständen selbst, können Aktionen ausgeführt werden. Aktionen können Ereignisse bei einem anzugebenden Zielobjekt auslösen. Hierbei kann der Benutzer Bezug auf Zielobjekt, Methode und Parameter des Methodenaufrufes nehmen.

Textuelle Beschreibungen können nur Zuständen und Zustandsübergängen zugeordnet werden. Eine Möglichkeit, auch Aktivitäten und Ereignisse mit Beschreibungen auszustatten, besteht nicht. Ebenfalls unmöglich ist die Modellierung einer Generalisierung von Ereignissen.

Dynamisches Objektverhalten - Objektdynamik, S. 39

1.	+
2.	+
3.	+
4.	+
5.	-
6.	-

d) Dynamisches Systemverhalten - Systemdynamik

Eine Aggregation von Zuständen nebenläufiger Objekte ist nicht möglich. Beim manipulieren von Sequenzdiagrammen kann auf die Methoden einer Klasse zurückgegriffen werden.

Dynamisches Systemverhalten - Systemdynamik, S. 39

1.	-
2.	+

e) Zusammenhang zwischen den Modellen

Rational Rose verbindet Zustandsdiagramme eindeutig mit Klassen. Ist in der Klassendiagrammansicht eine Klasse ausgewählt, gelangt der Benutzer über das Menü/Kontextmenü zu dem dazugehörigen Zustandsdiagramm, bzw. kann er dieses anlegen. Auch im Browser werden Zustandsdiagramme, im Gegensatz zu anderen Diagrammen nicht zusammengefaßt dargestellt, sondern sind der jeweiligen Klasse zugeordnet. Besitzt die Klasse, zu der ein Zustandsdiagramm gehört, Unterklassen, so ist jeder dieser Unterklassen ein eigenes Zustandsdiagramm zugeordnet. Das dynamische Verhalten einer Klasse wird nicht vererbt.

Ereignisse aus dem Zustandsdiagramm, die zu Zustandsübergängen führen, können Methoden des statischen Objektmodells zugeordnet werden. Die Möglichkeit, bei der Modellierung von Zuständen auf die Attribute einer Klasse Bezug zu nehmen, besteht nicht.

Zusammenhang zwischen den Modellen, S. 39

1.	+
2.	-
3.	+
4.	-

f) Abstraktionsebenen / Sichten

Der Detaillierungsgrad der Darstellung im Klassendiagramm kann vom Benutzer variiert werden. Dies geschieht entweder im Optionen-Dialog, bei den Diagrammeinstellungen, für die gerade gewählte Klasse über das Kontextmenü oder im Falle mehrerer gewählter Klassen über das Menü *bearbeiten*. Im ersteren Fall werden die Einstellungen global vorgenommen und beziehen sich auf alle zukünftig generierten Diagramme. Soll der Detaillierungsgrad der Darstellung bereits bestehender Notationselemente variiert werden, so müssen diese angewählt und die Veränderungen über Menü/Kontextmenü vorgenommen werden. Verschiedenen Diagrammen per Voreinstellung unterschiedliche Detaillierungsniveaus zuzuweisen ist nicht möglich. Die Einstellmöglichkeiten umfassen das Ein- bzw. Ausblenden von:

Abstraktionsebenen / Sichten, S. 40

1.	+
2.	-
3.	-
4.	-
5.	+

- ❑ Stereotypen
- ❑ Symbol für Sichtbarkeit
- ❑ Signaturen der Operationen
- ❑ Attribute
- ❑ Operationen

Das Programm ist nicht in der Lage Attribute oder Operationen hinsichtlich ihrer Sichtbarkeit ein- bzw. auszublenden.

Der Benutzer hat die Möglichkeit, bestimmte Elemente auszuwählen, die in jedem Fall angezeigt werden, auch wenn Details dieser Art ausgeblendet sind.

In der Zustandsdiagrammansicht kann der Benutzer festlegen, ob der Inhalt von Superzuständen dargestellt werden soll oder nicht.

g) Vorgangsreihenfolge bei der Entwicklung

Vorgangsreihenfolge bei der Entwicklung, S. 40

1.	+
2.	+
3.	+
4.	-

Rational Rose zwingt den Benutzer nicht, eine bestimmte Reihenfolge beim Anlegen eines Modells einzuhalten. Es existiert kein Vorgehensmodell, das den User leitet, bzw. an das er sich halten muß. Allerdings bietet der Hersteller ein Zusatzwerkzeug namens *Rational Unified Process* an, das ein solches Vorgehensmodell definiert und mit Rose zusammenarbeitet.

Werden Notationselemente neu angelegt, werden sie in jedem Fall sofort in das Repository übernommen. Dies geschieht, egal ob der Benutzer das neue Element in ein Diagramm gezeichnet, direkt im Repository definiert oder per Reverse-Engineering eingeführt hat. Neue Elemente benötigen bei ihrer Erzeugung nicht mehr als ihren Namen. Details wie Datentyp, Sichtbarkeit, Rückgabetyp oder Parameterliste können zu einem beliebigen späteren Zeitpunkt angegeben werden.

Alle Daten eines Modells werden in einem Data-Dictionary gespeichert. Diagramme stellen eine Sicht auf einen Teilbereich der Informationen des Data-Dictionarys dar. Es ist daher unmöglich, daß Notationselemente in einem Diagramm existieren, die jedoch nicht im Datenstamm des Systems vorkommen. Andererseits ist es sehr wohl möglich, daß Elemente, die im Data-Dictionary vorhanden sind, in keinem Diagramm vorkommen. Dies ist zum Beispiel der Fall, wenn der Benutzer Notationselemente direkt im Data-Dictionary anlegt, oder ein Element aus einem Diagramm, nicht jedoch aus dem System gelöscht wurde. Solche Elemente können in ein Diagramm aufgenommen werden. Dies geschieht entweder per Drag & Drop aus dem Browserfenster heraus, oder indem ein neues Element gleicher Art mit gleichem Namen angelegt wird.

h) Unterstützung bei der Methodenanwendung

Unterstützung bei der Methodenanwendung, S. 40

1.	+
2.	+
3.	+
4.	+
5.	-
6.	*0*

Das Tool wird mit einem *Methoden-Tutorial* ausgeliefert, das eine Einführung in die UML darstellt. Diese Einführung besteht aus mehreren Windows-Hilfedateien und ist sehr umfangreich. Auch in der kontextsensitiven Online-Hilfe von Rational Rose findet man einfache Erklärungen zur UML.

Es ist Literatur von Drittanbietern auf dem Markt, die sich explizit mit der Modellierung mit Rational Rose beschäftigt (etwa [Harm98] oder [Quat97]).

5.5.2 Funktionale Kriterien

a) Datenhaltung

Datenhaltung, S. 41

1.	+
2.	+
3.	+
4.	+
5.	*0*
6.	-
7.	*0*
8.	-

Rational Rose verfügt über ein zentrales Repository, in dem alle Informationen über das Modell gespeichert werden. Der Inhalt des Repositorys kann mit einem Browser eingesehen werden, welcher als Kindfenster oder gedockt an das Hauptfenster angezeigt werden kann. Im Browserfenster übernimmt ein Tree-Control die Anzeige der Daten. Daten, die in mehreren Diagrammen vorkommen, werden im Datenstamm nicht redundant gespeichert.

Der Benutzer kann die Modellinformationen im Browser genauso bearbeiten, wie dies in einer Diagrammansicht möglich wäre. Man gelangt über den Browser in den Eigenschaften-Dialog eines Elementes, kann Code generieren (Klasse, Package) und man hat selbstverständlich Zugriff auf Editierfunktionen wie Umbenennen, Löschen oder neu Anlegen. Notationselemente, die aus dem Repository gelöscht werden, werden auch aus allen Diagrammen entfernt, in denen sie vorkommen. Im Gegensatz dazu werden Elemente, die aus einem Diagramm entfernt werden, nicht automatisch aus dem Repository entfernt. Dies geschieht nur dann, wenn der Benutzer explizit den Befehl *Delete from Model* (Shortcut *Ctrl-D*) wählt.

Rational Rose bietet eine Schnittstelle, um Daten mit einem Microsoft Repository auszutauschen (Import/Export). Die Funktion *Print Specification* ist eigentlich dazu vorgesehen, Informationen über das Projekt auf einem Drucker auszugeben. Der Ausdruck kann allerdings in eine ASCII-Datei umgeleitet werden, was eine nützliche Exportfunktion darstellt. Außerdem speichert Rational Rose die Informationen zu Modellen als formatierten ASCII-Text. Mit relativ geringem Aufwand sollte es demnach gelingen, aus diesen Petal-Files (*.mdl) jede gewünschte Information zu extrahieren. Das Programm basiert nicht auf einem Datenbanksystem.

b) Browsing-Eigenschaften

Browsing-Eigenschaften, S. 41

1.	+
2.	-
3.	-
4.	-
5.	+
6.	+
7.	+
8.	-
9.	-
10.	+
11.	-

Das Programm verfügt über eine Suchfunktion, mit der sich beliebige Elemente eines Modells, egal welcher Art, leicht auffinden lassen. Nachdem ein entsprechender Suchbegriff eingegeben und die Suche angestoßen wurde, erhält der Benutzer in einer Liste alle gefundenen Elemente und deren Typ. Durch einen Doppelklick auf eines der Elemente dieser Liste öffnet Rational Rose das entsprechende Diagramm im Editor und zentriert die Ansicht auf das entsprechende Objekt. Macht dies keinen Sinn (z.B. Attribut, Operation), wird durch den Doppelklick der entsprechende Eigenschaftendialog geöffnet.

Eine zweite Möglichkeit, Elemente aufzufinden, ist die, im Browserfenster einen Buchstaben einzugeben. In diesem Fall wird das nächste Element mit entsprechendem Anfangsbuchstaben markiert. Weitere Funktionen, die den Benutzer bei der Suche nach bestimmten Notationselementen unterstützen, wie Suchfilter oder Suche nach bestimmten Ähnlichkeitskriterien, bietet das Programm nicht.

Um weitere Informationen über ein bestimmtes Notationselement zu erlangen, stellt das Werkzeug die Funktion *Show Usage* zur Verfügung. Der Benutzer erhält so eine Liste, in der jedes Vorkommen des Elementes aufscheint.

Die Elemente eines Modells können direkt im Browser manipuliert werden. Im Falle einer Namensänderung einer Klasse, eines Attributes oder einer Methode, kann der neue Name direkt im Browser eingegeben werden,

für alle anderen Änderungen muß der Eigenschaften-Dialog des Elementes geöffnet werden. In jedem Fall werden Änderungen in bestehenden Diagrammen sofort berücksichtigt.

Funktionen zur textuellen Auflistung von Klassenhierarchien oder zur Navigation in solchen existieren nicht.

Rational Rose erlaubt dem Benutzer, Dateien oder URLs, die der weiteren Beschreibung eines Modellierungselementes dienen, mit diesem zu verknüpfen. Ein entsprechender Link auf die Ressource wird dann im Browserfenster dem Element zugehörig dargestellt.

c) Navigation im Systemmodell

Navigation im Systemmodell, S. 42

1.	+
2.	+
3.	-
4.	+

Rational Rose erlaubt die Navigation zwischen den Ansichten eines Systems über das *Browse*-Menü. Hier werden zu jedem Diagrammtyp alle vorhandenen Diagramme angeführt und der Benutzer kann sie im Editor öffnen.

Zustandsdiagramme, die immer mit einer bestimmten Klasse verknüpft sind, kann der Benutzer der Einfachheit halber auch über das Kontextmenü der jeweiligen Klasse erreichen.

Packages lassen sich mit dem Menübefehl *Expand* oder mit einem Doppelklick öffnen und stellen ihren Inhalt in einem Diagramm dar.

Zwischen Kollaborations- und Sequenzdiagramm kann ebenfalls mittels Befehl aus dem *Browse*-Menü navigiert werden. Soll von einem der beiden Diagrammtypen zu seinem zugehörigen Gegenstück navigiert werden und dieses existiert noch nicht, wird es vom Werkzeug automatisch generiert.

d) Konsistenzprüfung / Integrität

Konsistenzprüfung / Integrität, S. 42

1.	+
2.	*0*
3.	+
4.	+
5.	+
6.	-
7.	+
8.	-
9.	+

Das Werkzeug verfügt über eine Konsistenzprüfung, die nicht eigens ausgelöst wird, sondern ständig aktiv ist und das System überwacht, um Unregelmäßigkeiten erst gar nicht zuzulassen. Zum Beispiel bricht dieser Mechanismus das Anlegen einer rekursiven Vererbungsbeziehung mit einer entsprechenden Fehlermeldung ab. Beziehungen, die ins Leere gehen, können nicht angelegt werden, ebensowenig unterschiedliche Klassen mit gleichem Namen. Namenlose Attribute und Methoden werden nicht zugelassen, wohingegen Klassen oder Beziehungen ohne Namen vom System toleriert werden.

Rational Rose verfügt auch über zwei Funktionen, um ein Modell zusätzlich zu überprüfen. Zum einen ist das die Funktion *Check-Model*, die das System nach ungültigen Referenzen durchsucht und diese in einem Log-Fenster auflistet. Die zweite derartige Funktion, *Show Access Violation*, überprüft die Referenzen auf Klassen aus fremden Paketen auf deren Zulässigkeit.

Wird ein Modellierungselement aus einem Diagramm gelöscht, bleibt es im System dennoch vorhanden. Erst wenn es explizit im Repository angewählt und gelöscht wird, oder beim Löschen aus dem Diagramm der Befehl *Delete From Model* verwendet wird, wird das Element komplett entfernt. Als Schutz vor unbeabsichtigtem Datenverlust kann ein Notationselement mit der *Entfernen*-Taste nicht komplett aus dem System entfernt werden, obwohl diese Taste als Shortcut für alle Löschoperationen vorgesehen ist. *Delete From Model* ist über die Tastenkombination *Steuerung-D* zu erreichen, bei der die Gefahr einer unabsichtlichen Betätigung nicht gegeben ist.

Notationselemente, die im Datenstamm existieren, jedoch in keinem Diagramm vorkommen, lassen sich leider nicht übersichtlich darstellen.

e) Report-Ausgabe

Report-Ausgabe, S. 43

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	0
8.	-
9.	-
10.	+

Die Report-Ausgabe erfolgt standardmäßig in MS-Word. Fordert der Benutzer einen automatisch generierten Report an, kann er in äußerst bescheidenem Maß dessen Inhalt und Detailtiefe einstellen. Der so erzeugte Report muß leider als weder hübsch, noch übersichtlich, noch umfangreich bezeichnet werden.

Die Reportausgabe von Rational Rose kann verbessert werden, indem man die standardmäßig mitgelieferten Skripts bearbeitet oder ersetzt. Solche Skripts sind im Internet verfügbar können aber auch von einem versierten Benutzer selbst erstellt werden.

Der Hersteller des Werkzeuges bietet auch ein Programm namens Rational SoDA an, das die automatische Dokumentationserzeugung wesentlich verbessern soll. Das Programm arbeitet mit allen Werkzeugen aus der Rational Suite zusammen und ist daher in der Lage, Informationen aus vielen verschiedenen Quellen, wie Test- oder Anforderungsdaten, in die Dokumentation einzubeziehen.

Als zusätzliches Feature bietet Rational Rose die Möglichkeit, mittels dem sogenannten Web Publisher, Reports zu einem Modell im HTML-Format zu generieren. Dies dient in erster Linie dazu, Modelle auch Leuten zugänglich zu machen, die über keine Rational Rose Lizenz verfügen. Im Gegensatz zu Reports mit reinen Textverarbeitungssystemen sind die HTML-Dokumente navigierbar und vermitteln so einen besseren Eindruck von dem zugrundeliegenden Modell.

f) Datenbankschema-Ausgabe

Datenbankschema-Ausgabe, S. 43

1.	+
2.	-
3.	-
4.	-
5.	-
6.	0

Rational Rose ist in der Lage, Datenbankschemata für relationale Datenbanksysteme auf Basis der Informationen des Repository zu erstellen. Hierzu wird ein Mapping-Interface für SQL-IDL zur Verfügung gestellt. Die explizite Benennung von Schlüsselattributen, oder die Angabe von Domains zu Attributen ist nicht möglich.

Die Darstellung von Entity-Relationship-Diagrammen wird ebensowenig unterstützt wie die explizite Generierung eines objektorientierten Datenbankschemas.

Der vom Werkzeug generierte DDL-Code enthält nur die wichtigsten Strukturen des statischen Modells. Informationen wie Multiplizität werden nicht berücksichtigt.

g) Diagramm-Ausgabe

Diagramm-Ausgabe, S. 44

1.	+
2.	+
3.	-
4.	0
5.	+
6.	-

Rational Rose unterstützt alle Drucker, die mit MS-Windows zusammenarbeiten. Mittels dem Befehl *Print setup* wird der Eigenschaften-Dialog für den Druckertreiber angezeigt, wie dies bei Windows-Programmen üblich ist. Über den Druckertreiber kann auch die Ausrichtung des Blattes (Hoch-/ Querformat) eingestellt werden.

Der Ausdruck entspricht fast genau der Anzeige auf dem Bildschirm, lediglich die Icons, welche die Sichtbarkeit darstellen, werden nicht ausgedruckt. Die Qualität kann als sehr gut bezeichnet werden. Das Programm stellt den Benutzer vor die Wahl, ein Diagramm entweder an die Blattgröße anzupassen, oder in Originalgröße zu drucken. Einen Skalierungsfaktor für die Größe des Ausdrucks anzugeben, ist nicht möglich. Wird der Druck in Originalgröße gewählt, verteilt Rose das Diagramm gegebenenfalls auf mehrere Druckblätter, die zu einer Gesamtgrafik zusammengefügt werden können. Leider fehlt hierbei eine Vorschau-Funktion, die vor dem mitunter lange

dauernden Druckvorgang anzeigt, wie das Diagramm auf die einzelnen Blätter verteilt wird. Der Benutzer weiß im vorhinein nicht einmal, auf wieviele Blätter der Druck verteilt wird.

Die Qualität des Ausdrucks kann lediglich variiert werden, wenn der Druckertreiber dies zuläßt. Das Programm ist dazu nicht in der Lage.

Da der Ausdruck der Darstellung auf dem Bildschirm entspricht, kann die Detailtiefe über die Optionen des Grafikeditors verändert werden.

h) Offenheit - Import / Export

Offenheit - Import / Export, S. 44

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	+
8.	+

Der Toolhersteller Rational Software Inc. bietet eine Reihe zusätzlicher Programme an, die mit Rational Rose zusammenarbeiten.

- *Rational RequisitePro* unterstützt den Benutzer beim Anforderungs-Management.
- *Rational TeamTest* ist ein Zusatztool, das aus den Anwendungsfällen eines Modells Testanforderungen ableitet. Diese Integration von Anwendungsfallmodellierung und Testplanung verbessert die Effizienz der Test-Teams und sorgt so in weiterer Folge für ausgereiftere Endprodukte.
- *Rational ClearCase* ist Zusatzsoftware, die Rational Rose Multi-User- und Multi-Projekt-fähig macht. ClearCase fungiert in diesem Fall als gemeinsames Repository für Modell- und Source-Code-Information.
- *Rational SoDA* stellt erweiterte Möglichkeiten der automatischen Dokumenterstellung zur Verfügung.
- *Rational Unified Process* vereinheitlicht und koordiniert den Softwareentwicklungsprozeß

Rational Rose arbeitet mit einer großen Zahl an Programmen von Drittanbietern aus den Bereichen Versionsmanagement, Textverarbeitung, Softwareentwicklung und Datenbankmanagement zusammen.

Das Programm nützt die in Windows-Programmen übliche Form des Datenaustauschs mit anderen Programmen über die Zwischenablage. Dies gilt für Text genauso wie für grafische Elemente aus dem Diagrammeditor. Die Befehle Cut, Copy und Paste sind verfügbar und es sind hierfür auch die gewohnten Tastatur-Shortcuts vorhanden.

i) Code-Generierung

Code-Generierung, S. 45

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	+
8.	*0*
9.	-
10.	+
11.	-

Rational Rose unterstützt die automatische Codegenerierung für mehrere Zielsprachen. Hierzu gehören C++, Java und Ada, aber auch Programmiersprachen der vierten Generation wie Visual Basic. Ebenso generiert das Tool IDL-Code für Anwendungen in CORBA und DDL-Code für Datenbankanwendungen.

Die Codegenerierung wird für eine oder mehrere Klassen oder Pakete ausgeführt. Bei der getesteten Codegenerierung in C++ erzeugt der Codegenerator für jede Klasse eine Header- und eine Code-Datei. Während die Header-Datei, welche die Deklarationen enthält, meist schon komplett ist, werden in der Code-Datei nur die leeren Funktionsrümpfe erzeugt, die händisch ergänzt werden müssen. Jedes Attribut wird mit dem entsprechenden Datentyp umgesetzt. Standardmäßig werden Beziehungen je nach der entsprechenden Multiplizität als (Pointer-)Variable oder als verkettete Liste deklariert. Die festgelegten Methoden der Klasse werden samt Parameterliste und Rückgabetyp erzeugt. Bei Attributen und Methoden wird auch die Sichtbarkeit im Code umgesetzt.

Der Codegenerator legt automatisch Konstruktoren, Destruktor und Vergleichsoperatoren an. Außerdem werden Zugriffsoperationen (*Get*, *Set*) für jedes Attribut der Klasse, sowie deren Beziehungen, erzeugt.

Der Code, den Rational Rose erzeugt ist mit Kommentarzeilen überfrachtet. Nur ein kleiner Teil dieser Kommentare kann vom Benutzer interpretiert werden und dient so dem Verständnis. Der Rest besteht aus Zahlenreihen und Variablen, die das Programm selbst verarbeitet.

Jeder erzeugte Funktionsrumpf enthält zwei Kommentarzeilen, die Beginn und Ende des Funktionskörpers festlegen. Code, den der Benutzer zwischen diese beiden Kommentarzeilen einfügt, wird vom Werkzeug erhalten. Auch bei nochmaligem Generieren des Codes ohne vorheriges Reverse Engineering wird solcherart hinzugefügter Code erhalten.

Rational Rose erlaubt dem Benutzer eine große Anzahl von Einstellungen für die automatische Codegenerierung vorzunehmen. Dies betrifft vor allem Entscheidungen über das automatische generieren von verschiedenen Konstruktoren oder Zuweisungsoperatoren. Außerdem können automatisch Abfrage- und Zuweisungsmethoden für Attribute, oder Methoden zur Abfrage von Relationen erzeugt werden. Diese Einstellungen können für das gesamte Modell im Optionen-Dialog, oder für ein bestimmtes Modellelement alleine (Klasse, Attribut, Methode) in dessen Eigenschafts-Dialog festgelegt werden.

Codeteile schon während des Modellierens den Methoden zuzufügen, ist nicht vorgesehen.

j) Reverse-Engineering

Reverse-Engineering, S. 45

1.	*+*
2.	*+*
3.	*-*

Rational Rose besitzt zum Zweck des Reverse-Engineerings von C++ Code ein zusätzliches Programm namens *C++ Analyzer*. Code, der in anderen Programmiersprachen wie Java oder Visual Basic vorliegt, wird beim Reverse-Engineering direkt von Rose bearbeitet.

Im C++ Analyzer können beliebige Quellcodedateien zu Projekten zusammengefaßt und analysiert werden. Nach dem Analysieren kann ein Projekt exportiert werden, wobei es dann als *.mdl Datei gespeichert und so von Rational Rose weiterverarbeitet werden kann. Sollen die Informationen aus dem Analyzer in ein bestehendes Rose-Projekt aufgenommen werden (z.B. beim Round-Trip Engineering), müssen die Informationen im Analyzer als *.red Datei gespeichert, und im Werkzeug selbst über den Menübefehl *Update* importiert werden.

Auch Code, der nicht vom Programm selbst stammt, kann auf diese Weise eingelesen werden. In einem solchen Fall wird jedoch Code, der über eine Deklaration hinausgeht, ignoriert. Wird Code, der von Rose stammt, händisch wie vorgesehen erweitert (im *body*-Bereich) und danach analysiert, bleibt der manuell angelegte Teil erhalten.

Test / Verifikation, S. 45

1.	*0*
2.	*-*

k) Test / Verifikation

Es ist möglich, testunterstützende Software zusammen mit Rational Rose zu verwenden. Das Programm ist in die Rational Suite eingebunden, in der professionelle Tools für Requirement Engineering, Testen, Change Request Management und Dokumentation enthalten sind.

Maßzahlen und Statistiken, S. 46

1.	*-*
2.	*-*
3.	*-*
4.	*-*

l) Maßzahlen und Statistiken

Es gibt Zusatzwerkzeuge, die als Add-Ins in Rational Rose installiert werden können, und die Maßzahlen und Statistiken basierend auf den Daten des Re-

pository generieren. Diese Werkzeuge standen für die Evaluation jedoch nicht zur Verfügung.

m) Multi-User - Eigenschaften

Multi-User - Eigenschaften, S. 46

1.	+
2.	-
3.	-
4.	+
5.	-
6.	-
7.	-

Das Programm besitzt Features, die es einem Team von Entwicklern ermöglicht, gemeinsam über ein Netzwerk an einem Projekt zu arbeiten. Hierbei wird das Modell in einzelne Pakete, die *Controlled-Units*, zerlegt. Jeder Entwickler in der Multi-User-Umgebung arbeitet jetzt in seinem eigenen, privaten Arbeitsbereich, der eine individuelle Ansicht des kompletten Modells darstellt. Er kann nun parallel zu anderen Benutzern an Problemen arbeiten, ohne diese zu beeinflussen. Änderungen in einem privaten Arbeitsbereich werden erst öffentlich, und somit den anderen Entwicklern zugänglich, wenn es ausdrücklich gewünscht wird. Dies geschieht über ein *Configurations-Management und Versionskontrollsystem* (CMVC). Rational Rose arbeitet mit gängigen CMVC-Systemen zusammen (z.B. Rational ClearCase und Microsoft SourceSafe).

Kommunikation mit anderen Projektmitarbeitern aus dem Tool heraus ist nicht möglich.

n) Security

Security, S. 46

1.	-
2.	-
3.	-
4.	-
5.	-
6.	-
7.	-
8.	-
9.	+
10.	+
11.	+

Rational Rose selbst besitzt keine Benutzerverwaltung. Da jedoch Controlled Units in eigenen Dateien verwaltet werden, können auf diese vom Betriebssystem aus (oder durch ein Konfigurations-Management-Programm) verschiedene Benutzerrechte vergeben werden.

Da das Programm jedes Projekt in einer eigenen Datei speichert, stellt ein Backup kein Problem dar. Rational Rose basiert nicht auf einem Datenbanksystem, ein Recovery-Mechanismus, der im Fehlerfall die Konsistenz gewährleisten, ist daher nicht vorhanden.

Kritische Aktionen, wie das Löschen von Modellierungsinformationen, muß der Benutzer gesondert bestätigen, bevor sie vom Programm ausgeführt werden. Rose besitzt eine Undo-Funktion, mit der Aktionen rückgängig gemacht werden können.

o) Multi-Projekt-Eigenschaften / Versionsmanagement

Multi-Projekt-Eigenschaften / Versionsmanagement, S. 47

1.	+
2.	+
3.	-
4.	+
5.	-

Die zeitlich parallele Abwicklung mehrerer Projekte ist mit Rational Rose kein Problem, da jedes Projekt als einzelne Datei gespeichert wird. Soll ein anderes Projekt geladen werden, wählt der Benutzer einfach den Befehl *open* aus dem Datei-Menü aus, woraufhin das derzeit aktuelle Projekt geschlossen und eventuell gespeichert wird, und das vom Anwender gewünschte Modell geladen wird.

Ein Austausch von Modellierungsinformationen zwischen unterschiedlichen Projekten ist auf zwei Arten möglich. Zum einen können Modellelemente mittels Copy-Befehl in die Zwischenablage übertragen, und nach dem Laden des neuen Projektes in dieses eingefügt werden. Die zweite Möglichkeit ist, Modellierungselemente aus einem Projekt zu exportieren, dabei werden sie in einer sogenannten Petal-Datei (*.ptl) abgelegt, und in ein anderes zu importieren. Eine Bezugnahme auf Modellelemente eines anderen Projektes, ohne diese zu kopieren, ist nicht möglich.

Rational Rose ist nicht selbst mit einem Versionsmanagement-Werkzeug ausgestattet, arbeitet jedoch mit selbständigen Programmen zur Versionskontrolle zusammen. Das Werkzeug ist auf seine Fähigkeiten zur Zusammenarbeit mit Rational ClearCase und Microsoft Visual SourceSafe getestet

worden. Es wird hierbei die SCC-API von Microsoft verwendet, die Standardbibliothek für Versionskontrollsysteme. Aus diesem Grund ist es wahrscheinlich, daß Rose auch mit anderen Werkzeugen dieser Art kooperiert, dies wurde jedoch nicht getestet.

Das Programm besitzt eine Funktion namens *Visual Differencing*, mit deren Hilfe sehr übersichtlich Unterschiede zwischen dem aktuellen und einem Referenzmodell angezeigt werden können.

5.5.3 Ergonomie

a) Benutzeroberfläche

Benutzeroberfläche, S. 47

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	+

Rational Rose besitzt in der getesteten Windows-Version eine ansprechende, moderne und funktionelle Oberfläche. Ein Benutzer, der mit Windows-Programmen Erfahrung hat, wird sich mit dem Tool schnell zurechtfinden. Der Browser ist in einem Fenster mit variabler Größe untergebracht, das an die Hauptansicht gedockt ist, die Struktur des Repositorys wird in Form eines Tree-Controls dargestellt, tabellarische Daten werden in windowstypischen List-Controls mit variabler Spaltenbreite und Sortierfunktion ausgegeben, und in jedem Bereich des Programms wird Gebrauch von Kontextmenüs gemacht.

Die Menüaufteilung von Rational Rose entspricht dem Windows-Standard (*File, Edit, ..., Window, ?*). Es stehen die üblichen Funktionen der Zwischenablage (Cut, Copy, Paste) zur Verfügung und diese sind auch wie gewöhnlich zu erreichen (Menü *Edit* bzw. *Strg-x/c/v*).

Die Dialoge des Programms sind teilweise sehr überladen und in einige Unterseiten aufgeteilt. In Anbetracht ihres Umfanges sind sie jedoch relativ angenehm zu bedienen. Die Bezeichnungen der Menüeinträge sind selbsterklärend und es lassen sich immer nur jene Befehle aus den Menüs/Kontextmenüs wählen, die momentan verfügbar sind, alle anderen werden grau dargestellt.

Um den Benutzer darüber in Kenntnis zu setzen, daß das Programm an einer länger dauernden Aktion arbeitet, wird die Mauszeigergrafik verändert (Sanduhr). Wird zum Beispiel ein Projekt geladen, wird der Fortschritt des Ladevorganges zusätzlich als Balken in der Statuszeile dargestellt.

Rational Rose wurde hinsichtlich seiner Kompatibilität mit der Windows-Plattform von Microsoft getestet.

b) Tool-Dokumentation

Tool-Dokumentation, S. 48

1.	+
2.	+
3.	-
4.	-
5.	-
6.	+
7.	-
8.	*0*
9.	+
10.	+
11.	+
12.	+

Rational Rose wird mit einer ausführlichen gedruckten Dokumentation ausgeliefert, die aus mehreren Bänden verschiedener Stärke besteht. Dies sind ein allgemeines Handbuch, Informationen zum Round-Trip-Engineering mit C++, Java und Visual Basic, ein Handbuch, welches das Erweiterungsinterface REI beschreibt und gedruckte Release Notes. Die Release Notes enthalten letzte Informationen zu Rose und eine Installationsanleitung. Die Handbücher besitzen ein Stichwortverzeichnis.

Zusätzlich erhält der Rose Käufer das Buch *Visual Modelling with Rational Rose And UML* aus dem Addison-Wesley Verlag [Quat97].

Ein Online Handbuch liegt nur in Form eines kontextsensitiven Hilfesystems vor, das sehr umfangreich und informativ ist. Extra-Handbücher für versierte Benutzer bzw. Anfänger sind nicht vorhanden.

c) Usability

Usability, S. 48

1.	0
2.	0
3.	+
4.	-
5.	+
6.	+
7.	+
8.	+
9.	+
10.	+
11.	-

Längere Aktionen wie die Codegenerierung, oder das Laden eines Projektes lassen sich nicht vom Benutzer unterbrechen.

Im Fehlerfall wird der Benutzer zwar entweder durch einen Dialog, oder einen Eintrag im Log-Fenster über den Fehler informiert, das Aufrufen von Hilfe zu diesem Problem ist jedoch nicht vorgesehen.

Tastatur-Shortcuts stehen dem Benutzer in großer Zahl zur Verfügung. Neben den Standard Shortcuts für Dateioperationen (neu, laden, speichern) und Bearbeitungsfunktionen (Cut, Copy, Paste, Undo, Redo, Find,...) stehen solche auch für viele Befehle im *View*-Menü (*Zoom, Fit in Window, Refresh*,...) und im *Browse*-Menü (*Specification, Previous Diagram, Expand,*...) bereit. Funktionstasten finden zwar als Shortcuts Verwendung, können jedoch nicht extra belegt werden.

Das Programm stellt in jeder Situation ein dementsprechend angepaßtes Kontextmenü bereit, das über die rechte Maustaste zu erreichen ist. Ungültige Operationen, soweit sie von Rose überhaupt zugelassen werden, versetzen das Programm keineswegs in einen nicht definierten Zustand, sondern werden abgefangen und entsprechend behandelt (Fehlermeldung).

Rational Rose verfügt über mehrere Werkzeugleisten. Die Standardleiste, die horizontal unter der Menüleiste des Hauptfensters angebracht ist stellt allgemeine Befehle zur Veränderung der Ansicht oder zum Navigieren im Modell bereit. Die verschiedenen Leisten der Diagrammeditoren sind vertikal neben dem Editorfenster angebracht und beinhalten die Zeichentools. Alle Werkzeugleisten können, wie dies bei der jüngsten Generation von Windows-Programmen üblich ist, per Drag&Drop aus ihrer Verankerung gelöst werden, wodurch sie in einem eigenen Kindfenster, frei beweglich dargestellt werden. Rose erlaubt es, die Werkzeugleisten auf einfachstem Weg über das Optionen-Menü den Bedürfnissen des Benutzers anzupassen.

In einer Statuszeile unterhalb des Hauptfensters gibt das Programm Informationen an den Benutzer weiter. Innerhalb des Hauptfensters können beliebig viele Client-Fenster angelegt werden, die verschiedene Sichten auf das gerade geladene Projekt darstellen.

Eine Mehrfachauswahl ist in den meisten Situationen durch gedrückt halten der *Shift*-Taste möglich.

Rational Rose ist nur in englischer Sprache erhältlich.

d) Hilfefunktionen

Hilfefunktionen, S. 49

1.	+
2.	+
3.	+
4.	+

Rose besitzt ein umfangreiches Hilfesystem, das auf Windows-Help-Dateien beruht. In viele Dialoge ist ein Hilfe-Button eingebaut, der die Hilfedatei aufruft, und die richtige Seite anwählt. Die Hilfedatei enthält viele Links auf zugehörige Themen, was das Nachschlagen effizient gestaltet. Wie in jedem Windows-Hilfesystem kann auch hier nach Stichworten gesucht werden.

Rational Rose bietet Hot-Help an. Wird der Mauszeiger über einen Button in der Werkzeugleiste, oder ein Symbol im Klassendiagramm bewegt, erscheint ein kleines Textfenster mit Informationen darüber.

e) Graphikeditor / Darstellung

Der Graphikeditor von Rational Rose stellt, abgesehen von einigen Ausnahmen, wie zum Beispiel beim Sequenz- oder Kollaborationsdiagramm, alle Notationselemente UML-konform dar. Abgesehen von der generellen Umstellung zwischen den Notationsarten UML, OMT und Booch gibt es keine Möglichkeit, das Aussehen grafischer Elemente zu verändern, Abgesehen

von der Einbindung von Bitmaps für stereotypisierte Klassen. Ebenfalls nicht vorgesehen ist das Zeichnen freier Formen in ein Diagramm, um so eventuell dem Tool unbekannte Notationselemente nachzubilden. Lediglich einfacher Text läßt sich zusätzlich in einem Diagramm an beliebiger Stelle eintragen.

Graphikeditor / Darstellung, S. 49

1.	-
2.	-
3.	0
4.	+
5.	-
6.	+
7.	-
8.	+
9.	+
10.	+
11.	0
12.	-
13.	+
14.	-
15.	+

Die Darstellung der Diagramme am Bildschirm entspricht genau dem späteren Ausdruck, mit Ausnahme der Icons für die Sichtbarkeit, die nicht mitgedruckt werden. Es sind viele Möglichkeiten vorhanden, um Farbe und Beschriftung von Notationselementen anzupassen. Der Benutzer kann für die Beschriftung aus allen installierten Schriftarten einen Font wählen und zusätzlich die Schriftgröße und -farbe festlegen. Auch die Linienfarbe der Diagrammelemente kann in fast allen Fällen variiert werden. Ausgefüllte Elemente lassen zusätzlich die Spezifikation einer Füllfarbe zu.

Die Ansicht des Grafikeditors kann in kleinen Stufen vergrößert und verkleinert werden. Auch eine Funktion, um das Diagramm der Größe nach in das aktuelle Fenster einzupassen, ist vorhanden. Symbole und Textelemente im Diagramm können nicht rotiert werden.

Die Größe der Notationselemente kann vom Benutzer per Drag&Drop festgelegt werden. Wird ein Element neu angelegt oder der Name des Elementes verändert, paßt Rational Rose die Größe des Elementes an die Länge des Bezeichners an. Ein Zeilenumbruch innerhalb einer Element-Beschriftung kann nicht bei allen Elementen dargestellt werden. Ist zum Beispiel ein Klassensymbol zu schmal, um den gesamten Klassennamen darin anzuzeigen, wird dieser nur zum Teil dargestellt.

Um das Layout eines Diagrammes zu verbessern, können in bestehende Relationen auch nachträglich Eckpunkte eingefügt werden. Eine Funktion, mit der mehrere ausgewählte Elemente in irgendeiner Form angeordnet werden können (Alignment), fehlt. Das Layout der Diagramme wird bei der Speicherung berücksichtigt. Es existiert eine Funktion, mit welcher der Fensterinhalt neu gezeichnet werden kann.

f) Graphikeditor / Bedienung

Graphikeditor / Bedienung, S. 50

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	+
8.	+
9.	+
10.	+
11.	+
12.	+
13.	+
14.	+

Das Einzeichnen von Notationselementen in ein Diagramm funktioniert mittels Werkzeugen, die aus einer Iconleiste ausgewählt werden. Ist ein Werkzeug aktiv, kann das entsprechende Element mit einem Linksklick in das Diagramm eingezeichnet werden. Normalerweise steht ein gewähltes Werkzeug für eine Aktion zur Verfügung und deaktiviert sich danach selbst. Durch Drücken der Shift-Taste während des Einzeichnens wird dies verhindert und das Werkzeug bleibt aktiv.

Sofort nach dem Anlegen eines Elementes kann der Benutzer den Bezeichner eingeben. Das Programm vergibt automatisch einen Default-Namen (z.B. *NewClass*), markiert diesen und setzt den Cursor in das Namensfeld. Der voreingestellte Name kann so auf Wunsch einfach überschrieben werden. Im Klassendiagramm wird nach dem Anlegen einer neuen Klasse zusätzlich eine Listbox unter dem Namensfeld eingeblendet, aus welcher der Benutzer optional einen schon vorhandenen Klassennamen auswählen kann.

Der Bereich, der für das Zeichnen von Diagrammen zur Verfügung steht, ist unbegrenzt und kann, wie bei Windows-Programmen üblich, mittels Bildlaufleisten gescrollt werden. Die grafischen Elemente können auf diesem Zeichenbereich per Drag&Drop frei angeordnet werden. Es können mehrere Objekte selektiert werden, was durch Anklicken mit gehaltener Shift-Taste, oder durch Ziehen eines Rahmens mit der Maus geschieht. In einem solchen Fall wirken sich Editierbefehle, wie Verschieben oder Löschen auf alle gewählten Elemente aus.

Das Programm bietet die Möglichkeit, die grafischen Objekte der Diagramme auf einem Raster auszurichten (*Grid Alignment*). Im Optionen-Dialog bei den Diagrammeinstellungen kann die Größe des Rasters frei eingestellt werden, jedoch kann dieser nicht angezeigt werden.

Es ist praktisch immer möglich, das Diagramm händisch so zu manipulieren, daß alle Elemente am gewünschten Ort liegen. Eine irrtümlich initiierte Zeichenaktion läßt sich einfach durch einen Druck auf die rechte Maustaste abbrechen.

Rational Rose bietet sowohl eine Undo- als auch eine Redo-Funktion an. Die beiden Funktionen arbeiten jedoch oft nicht so, wie man dies erwarten oder sich wünschen würde. Zum Beispiel kann eine Größenänderung eines Symbols nicht mittels Undo rückgängig gemacht werden. Die Redo-Funktion ist manchmal nicht in der Lage, den zuletzt durch Undo rückgängig gemachten Arbeitsschritt wiederherzustellen.

Der Diagrammeditor stellt die üblichen Zwischenablagefunktionen wie Cut, Copy und Paste bereit. Es ist möglich, Elemente aus einem Diagramm auszuschneiden oder zu kopieren, und in ein anderes Diagramm einzufügen. Diese Funktionen können auch über die üblichen Tastatur-Shortcuts erreicht werden. Auch das Löschen von Elementen aus einem Diagramm ist mittels Shortcut (*Delete*) möglich.

g) Performance

Die Performance des Programmes in allen Situationen gab keinen Anlaß zur Kritik.

Performance, S. 51

1.	+
2.	+
3.	+
4.	+

h) Adaptivität

Rational Rose läßt sich umfassend an die Bedürfnisse des Benutzers anpassen. Hierfür wird das sogenannte *Rose Extensibility Interface* (REI) zur Verfügung gestellt. Dieses Interface erfüllt folgende Aufgaben:

- ❑ persönliche Anpassung der Benutzeroberfläche
- ❑ Automatisierung von Funktionen mittels Skripts
- ❑ Anbindung von Zusatztools durch einen Add-In-Manager

Rational Rose wird nicht mit dem Sourcecode ausgeliefert.

Adaptivität, S. 51

1.	-
2.	+
3.	+
4.	-
5.	+
6.	-

5.6 SELECT Enterprise 6.0e SR-4

SELECT Enterprise ist ein komponentenbasiertes Werkzeug zur Modellierung innerhalb großer, verteilter Entwicklungsumgebungen. Die kontrollierte Zusammenarbeit vieler Benutzer wird hierbei ebenso effizient unterstützt, wie die Wiederverwendung von Softwarekomponenten sowie die Datenhaltung, die über das OODBMS Softlab Enabler abgewickelt wird.

Die graphische Benutzeroberfläche orientiert sich sehr stark an den bekannten Windows-Styleguides, wirkt jedoch teilweise ein wenig überladen. Die Codegenerierung ist in mehreren Zielsprachen möglich, weiters steht eine Funktion zur automatisierten Synchronisation von Quellcode und zugrundeliegendem Modell zur Verfügung.

5.6.1 Methodenbasierte Kriterien

a) Modellierung der Objektstruktur - Objektstatik

Modellierung der Objektstruktur - Objektstatik, S. 38

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	+

Die Speicherung der Elemente und Informationen im Data-Dictionary erfolgt vollständig automatisiert im Hintergrund. Beim Anlegen neuer Elemente, wie auch bei der Änderung von Inhalten (z.B. Bezeichnern) wird automatisch auf Namenskonflikte überprüft. Wird etwa versucht, zwei Klassen mit demselben Bezeichner anzulegen, wird eine Fehlermeldung ausgegeben.

Die Reihenfolge der Komponenten kann selbst bestimmt werden, sie richtet sich jedoch prinzipiell nach der Reihenfolge der Eingabe, kann aber auch nachträglich verändert werden.

Die Parameter einer Methode können genau im *Property-Window* angegeben werden, sowohl ihr Typ als auch ihre Eigenschaft als Eingangs-, Ausgangs- oder Durchgangsparameter (*in*, *out*, *in/out*). Es besteht auch die Möglichkeit, textuell Vor- bzw. Nachbedingungen einer Methode einzutragen.

b) Modellierung der Systemstruktur - Systemstatik

Modellierung der Systemstruktur - Systemstatik, S. 39

1.	+
2.	+
3.	*0*
4.	*0*

Einmal erfaßte Beziehungen, gleich welcher Art, können nachträglich mittels Drag&Drop verändert werden.

Die Einteilung von Klassen erfolgt nur mittels Packages, bzw. mit Hilfe der folgenden vorgegebenen Stereotypen: Das Stereotyp <<business>> beschreibt im weiteren Sinn sämtliche Entity-Klassen eines Modells, <<user>> wird für Interface-Klassen verwendet und mit Hilfe von <<external>> werden Klassen beschrieben, die außerhalb des aktuellen Modells existieren und nur referenziert werden können.

OCL (*Object Constraint Language*)-Annotationen werden nicht direkt unterstützt, sondern müssen vom Benutzer textuell eingegeben werden und werden vom Tool nicht interpretiert.

c) Dynamisches Objektverhalten - Objektdynamik

Dynamisches Objektverhalten - Objektdynamik, S. 39

1.	+
2.	*0*
3.	*0*
4.	+
5.	+
6.	-

Zustandsdiagramme sind zwingend eindeutig mit einer Klasse verbunden. Dies wird dadurch erreicht, daß zur Erstellung eines neuen Zustandsdiagramms im Kontextmenü der Klasse der Punkt *New - State Diagram* ausgewählt wird und somit ein neues Diagramm angelegt wird.

Die Angabe von Bedingungen für Zustandsübergänge ist nur über textuelle Beschreibung möglich, es kann nicht direkt auf Attribute der Objekte

Bezug genommen werden. Eine Beschreibung von Aktivitäten u.a. ist ebenfalls möglich (Textfeld).

In Hinsicht auf die Generalisierung von Zuständen kann SELECT Enterprise sowohl UND- (*concurrent states*) als auch ODER- (*sequential states*) verfeinerte Zustände korrekt geschachtelt darstellen. Ereignisse selbst können nicht generalisiert werden.

d) Dynamisches Systemverhalten - Systemdynamik

Dynamisches Systemverhalten - Systemdynamik, S. 39

1.	0
2.	+

Beim Erstellen von Sequenzdiagrammen können vorhandene Methoden eines Objekts ausgewählt werden, es besteht auch die Möglichkeit, für eine Nachricht aktuelle Werte der Methoden-Parameter anzugeben (Kontextmenü *Instance Properties*). Auf Attribute einer Klasse kann jedoch nicht Bezug genommen werden.

Wird im Zuge der Kontrollflußmodellierung im Sequenzdiagramm eine neue Operation eingetragen, kann diese sofort näher spezifiziert werden (Bezeichner, Rückgabewert, Parameter etc.) und wird automatisch in der Datenbasis bzw. im Klassendiagramm übernommen.

e) Zusammenhang zwischen den Modellen

Zusammenhang zwischen den Modellen, S. 39

1.	+
2.	-
3.	0
4.	0

Ein Zustandsdiagramm ist umkehrbar eindeutig mit einer Klasse verbunden (siehe *Dynamisches Objektverhalten - Objektdynamik*, S. 114), das Zustandsdiagramm einer Klasse wird innerhalb einer Generalisierungshierarchie jedoch nicht an deren Unterklassen vererbt.

f) Abstraktionsebenen / Sichten

Abstraktionsebenen / Sichten, S. 40

1.	+
2.	-
3.	+
4.	-
5.	+

Innerhalb von Klassendiagrammen läßt sich der Detaillierungsgrad der Anzeige effizient über Kontextmenüs einstellen. Für Attribute wie auch für Operationen kann ausgewählt werden, welche der drei Typen *private*, *public*, bzw. *protected* angezeigt werden sollen. Leider ist es weder für Attribute, noch für Operationen möglich, die Signaturen (Typen, Parameter, u.a.) anzeigen zu lassen, es wird immer lediglich der Bezeichner (und auf Wunsch die Sichtbarkeit) dargestellt.

Verfeinerte Zustandsdiagramme lassen sich nicht in verschiedenen Detaillierungsstufen anzeigen.

Die Darstellungsweise läßt sich sowohl defaultmäßig festlegen, als auch für bestimmte Diagramme, wie auch für einzelne Elemente unterschiedlich.

g) Vorgangsreihenfolge bei der Entwicklung

Vorgangsreihenfolge bei der Entwicklung, S. 40

1.	+
2.	+
3.	+
4.	-

SELECT Enterprise zwingt den Benutzer nicht zu einer konkreten Vorgehensweise bei der Erstellung des Modells, es kann eine solche auch nicht im Tool vorgegeben werden. Es ist ohne weiteres möglich, Klassen nur mit ihrem Bezeichner anzulegen und die unterschiedlichen Informationen nachträglich zu verfeinern. Neu angelegte Elemente werden automatisch im Data-Dictionary gespeichert.

Außerdem besteht die Möglichkeit, Elemente (auch Beziehungen zwischen Elementen etc.) zunächst im Data-Dictionary einzutragen und später in einem Diagramm zu übernehmen. Dies erfolgt einfach mittels Drag&Drop aus dem Repository-Browser in das entsprechende Diagramm. Diese Funktion wird durchgängig und effizient unterstützt, so kann der Benutzer etwa eine Operation einer Klasse in ihrem zugehörigen Zustandsdiagramm in einen bestimmten Zustand ziehen. Daraufhin wird automatisch eine *do*-Aktion

für die entsprechende Operation generiert. So kann auch die Konsistenz des Modells problemlos sichergestellt werden.

h) Unterstützung bei der Methodenanwendung

Unterstützung bei der Methodenanwendung, S. 40

1.	+
2.	+
3.	+
4.	*0*
5.	*0*
6.	*0*

Bei der Beurteilung der UML-Methodenunterstützung ist zu berücksichtigen, daß SELECT Enterprise einen eigenen Zugang, bzw. eine eigene "Philosophie" in Hinsicht auf objektorientierte Modellierung von Geschäftsprozessen beinhaltet - der *SELECT Perspective approach to Component Based Development* (vgl. Übersichtsteil). Hierzu stellt das Tool umfangreiche Online-Tutorials (*Process Mentor*) zur Verfügung, in denen der gesamte Modellierungsprozeß, einschließlich der UML.Modellierung, ausführlich und anhand von Beispielen praxisnah beschrieben wird. So werden auch im Handbuch nicht ausschließlich die Konzepte der UML erläutert.

Die Diagrammelemente entsprechen weitgehend denen des UML 1.1 Standards, so daß entsprechende Basisliteratur ohne Probleme Verwendung finden kann.

Bei Verletzungen der Konsistenz werden lediglich die entsprechenden Fehlermeldungen ausgegeben, eine weiterführende Erläuterung bleibt aus.

5.6.2 Funktionale Kriterien

a) Datenhaltung

Datenhaltung, S. 41

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	+
8.	*0*

Die Datenhaltung ist eine der besonderen Stärken von SELECT Enterprise. Zur Anwendung kommt zur Zeit ein Produkt eines Drittherstellers, das Repository Enabler der Firma Softlab, das auf einer objektorientierten Datenbank aufbaut. Hiermit wird eine offene, vollständig skalierbare Architektur zur Verfügung gestellt, mit der Möglichkeit die Datenbasis sowohl zentral als auch im Intranet / Internet verteilt abzulegen. In Zukunft sollen neben Enabler auch das Microsoft Repository und UREP der Firma Unisys unterstützt werden, womit das verteilte Arbeiten einer großen Anzahl von Benutzern und den Austausch der Projektinformationen noch erweitert wird.

Modellinformationen, die in mehreren Diagrammen (wie auch in mehreren verteilten Projekten) vorkommen, werden nichtredundant gespeichert und können einfach im Data-Dictionary gewartet (hinzugefügt, geändert und gelöscht) werden.

Wie bereits beschrieben, können in einem bestimmten Diagramm noch nicht vorkommende, aber bereits angelegte Elemente einfach mittels Drag&Drop aus dem Repositorybrowser in das Diagramm eingefügt werden. Die Änderung eines Elements im Diagramm (oder im Repository) wirkt sich automatisch auch auf alle anderen Diagramme aus.

Das Löschen von Inhalten ist ebenfalls sehr gut implementiert. Es kann ausgewählt werden, ob eine Diagrammelement nur in diesem bestimmten Diagramm, oder komplett aus der Datenbasis gelöscht werden soll (*global delete*).

Das verwendete Repository Enabler bietet mächtige Schnittstellen zum Zugriff und zur Änderung der enthaltenen Daten, wie z.B. über C++ Klassenbibliotheken, Active/X-Controls, Java, SQL, u.a.

b) Browsing-Eigenschaften

Browsing-Eigenschaften, S. 41

1.	+
2.	-
3.	+
4.	+
5.	0
6.	+
7.	+
8.	0
9.	+
10.	0
11.	+

Die Suchfunktionen von SELECT Enterprise decken ein breites Spektrum an nützlichen Funktionen zum effizienten Auffinden von bestimmten Elementen, sowohl in der gesamten Datenbasis, als auch in einzelnen Diagrammen:

Im Repository-Browser, bzw. in verschiedenen Dialogfenstern genügt das Eingeben des Anfangsbuchstaben eines Elements, um eine entsprechende Auswahl anzuzeigen. Es kann mit dieser Funktion innerhalb jeder Ebene der Baumstruktur des Browsers gesucht werden.

Außerdem ist eine umfangreiche Suchfunktion über eine Suchmaske implementiert, in der die verschiedensten Filter angegeben werden können. Diese umfassen u.a. folgende:

- ❑ Angaben über den Bezeichner (Anfangsbuchstaben, enthaltene Phrase, Groß-/Kleinschreibung)
- ❑ Einschränkungen des Diagrammtyps, der das gesuchte Element enthält
- ❑ Datentypen
- ❑ Einschränkung der Suche auf unbenutzte Elemente
- ❑ letztes Änderungsdatum / Bearbeiter

Das Suchergebnis kann direkt weiter verwendet werden (Änderung der Eigenschaften etc.).

Eine bestehende Klassenhierarchie läßt sich bequem anzeigen, indem etwa die Superklasse einer Generalisation markiert wird und der Befehl *Browse Class* im Kontextmenü ausgewählt wird. Als Ergebnis erhält der Benutzer eine Übersicht über die Super- und Subklassen, die eigenen und die geerbten Attribute sowie Operationen. Diese können wiederum ausgewählt und näher spezifiziert (z.B. abstrakte Methoden redefiniert) werden.

Über spezielle Reports (deren Umfang und Inhalt bequem über ein Dialogfenster eingestellt werden kann) lassen sich genauere Informationen zu bestimmten Elementen abrufen.

c) Navigation im Systemmodell

Navigation im Systemmodell, S. 42

1.	+
2.	+
3.	0
4.	+

Die unterschiedlichen Diagramme eines Projekts lassen sich über das Browserfenster komfortabel direkt öffnen. Aber auch die Semantik der einzelnen Diagrammzusammenhänge läßt sich für die Navigation im Modell ausnutzen:

Um etwa zu einer Klasse das entsprechende Zustandsdiagramm aufzurufen, oder zu einem Use-Case Diagramm das zugehörige Sequenzdiagramm, genügt die Auswahl des entsprechenden Befehls im Kontextmenü eines Elements. Dies wird als *parent-child association* bezeichnet.

Darüber hinaus können Diagramme, sowie Elemente des Data-Dictionary untereinander auch mit eigenen Links versehen werden (*user-defined associations*).

d) Konsistenzprüfung / Integrität

Die "Online"-Konsistenzprüfung, die im Hintergrund von SELECT Enterprise die Eingaben des Benutzers und die Integrität des Modells überprüft, wie auch die separat anstoßbare Prüfung (*Consistency Check Report*), die das gesamte Modell bzw. die Datenbasis auf Konsistenz prüft, sind sehr umfangreich und genau.

Unter anderem wird der Versuch, folgende unrichtige Konstrukte anzulegen, abgefangen und verhindert:

- "namenlose" Elemente
- Beziehungen, Datenflüsse und Transitionen, die nicht mit zwei Elementen verbunden sind
- rekursive Generalisierungen
- mehrere Elemente mit gleichem Bezeichner (wobei SELECT Enterprise nicht *case-sensitive* arbeitet, so daß das Anlegen von zwei Attributen innerhalb einer Klasse mit den Bezeichnern "GebDatum" und "gebdatum" als unzulässig zurückgewiesen wird), u.ä.

Konsistenzprüfung / Integrität, S. 42

1.	+
2.	+
3.	+
4.	+
5.	+
6.	0
7.	+
8.	+
9.	+

Wie bereits beschrieben, lassen sich nicht genutzte Elemente, die nur in der Datenbasis, aber nicht in einem Diagramm vorkommen, mittels der Suchfunktion effizient anzeigen. Außerdem wird beim Löschen eines Elements aus einem Diagramm immer abgefragt, ob das Element nur aus dem Diagramm, oder auch aus der Datenbasis entfernt werden soll.

e) Report-Ausgabe

Report-Ausgabe, S. 43

1.	+
2.	+
3.	+
4.	+
5.	0
6.	+
7.	0
8.	0
9.	+
10.	+

SELECT Enterprise stellt umfangreiche Funktionen zur Generierung von Reports über die in Diagrammen, bzw. im Repository enthaltenen Elemente zur Verfügung.

Auf der einen Seite kommt der *Report Writer* zum Einsatz, mit dessen Hilfe textuelle Beschreibungen des Modells in einem Output-Fenster ausgegeben werden, die anschließend als Textdatei gespeichert werden können. Im Dialogfenster des Report-Writers hat der Benutzer die Möglichkeit, Angaben über den Umfang und den Inhalt des Reports anzugeben: Typen der beschriebenen Elemente, Diagrammtypen (Berücksichtigung sowohl der statischen als auch der dynamischen Modellaspekte), Konsistenzprüfung etc.

Weiters ist im Tool der sogenannte *Document Generator* integriert. Dieser stellt über eine Anbindung an MS-Word eine große Anzahl von Möglichkeiten zur Verfügung, Dokumentationen in veränderbarem Detaillierungsgrad über das aktuelle Projekt zu generieren. Hierbei wird etwa auch zwischen verschiedenen Stadien des Systementwicklungsprozesses unterschieden. So enthält eine Durchführbarkeitsstudie (*feasibility study*) Informationen über Projektplan, Use-Case Beschreibung, Aufwandsschätzungen etc., während Dokumente über die Analyse-, bzw. Designphase weitergehende Angaben beinhalten, wie statisches und dynamisches Modell, u.a. Die generierten Word-Dokumente sind sehr übersichtlich und ansprechend gestaltet (enthalten u.a. Abbildungen der Diagramme) und werden automatisch beim Ändern des Modells aktualisiert.

Abschließend können auch in einem Diagramm über einen Kontextmenüeintrag Berichte zur Konsistenz und über benutzte Elemente erzeugt werden.

f) Datenbankschema-Ausgabe

Datenbankschema-Ausgabe, S. 43

1.	+
2.	+
3.	0
4.	+
5.	0
6.	+

Zur Anbindung an die relationale Datenbankwelt stellt SELECT Enterprise einerseits die Möglichkeit zur Verfügung, sogenannte *Table Relationship* Diagramme zu erstellen, welche im wesentlichen eine Entity-Relationship-Modellierung darstellen. Es ist möglich, unabhängig vom eventuell bestehenden statischen UML-Modell eigene Tabellen mit den entsprechenden Informationsinhalten anzulegen.

Durch einen sogenannten *Storage-Mapping* Mechanismus können jedoch auch die im Objektmodell enthaltenen Informationen auf solche ER-Diagramme direkt abgebildet werden. Hierbei unterscheidet SELECT Enterprise strikt zwischen Datenmodell und UML-Modell, um den sogenannten *impe-*

dance mismatch zwischen den beiden Modellierungswelten zu umgehen. Angaben über Schlüsselattribute u.a. werden ebenfalls verarbeitet. Es ist weiters möglich, direkt SQL-Code (ANSI-SQL, Oracle etc.) zu erzeugen.

Auf der anderen Seite stehen Anbindungen an externe Datenmodellierungssysteme zur Verfügung, wie SELECT SE oder ERwin der Firma Platinum. Design-Informationen können zwischen den Werkzeugen bidirektional ausgetauscht werden, was einem Round-Trip-Engineering, ähnlich dem einer bestimmten Programmiersprache entspricht.

g) Diagramm-Ausgabe

Diagramm-Ausgabe, S. 44

1.	+
2.	-
3.	-
4.	+
5.	-
6.	+

SELECT Enterprise bietet eine komplett Windows-konforme Druckschnittstelle, die im Prinzip nur die von dem Betriebssystem bereitgestellten Einstellungsmöglichkeiten und Dialoge zur Verfügung stellt. Die Druckerausgabe arbeitet nach dem *What You See Is What You Get*-Prinzip. Auf Wunsch kann in Diagrammen direkt der Druckbereich angezeigt werden, auch eine *Print Preview* Funktion ist implementiert.

Möglichkeiten, den Ausdruck zu konfigurieren, bestehen im Papierformat, Größe, Hoch-/Querformat und Anpassung an die Seite. Weitere Möglichkeiten, wie kachelartiger Ausdruck oder Einstellung des Abstraktionsniveaus werden nicht unterstützt.

h) Offenheit - Import / Export

Offenheit - Import / Export, S. 44

1.	+
2.	+
3.	+
4.	-
5.	+
6.	-
7.	0
8.	0

SELECT Enterprise bietet Schnittstellen zu einer großen Anzahl von Werkzeugen, sowohl vom Hersteller selbst, wie der SELECT Component Manager zum Publizieren und Wiederverwenden von Komponenten, die in SELECT Enterprise erstellt wurden, oder SELECT Estimator, der zur Durchführung von Projektaufwandsschätzungen verwendet wird., wie auch zur Produkten von Drittherstellern, wie das IBM SanFrancisco Framework, oder Schnittstellen zu verschiedenen Entwicklungsumgebungen (Java, C++, Forté, u.a.). Außerdem ist eine offen gelegte OLE-Schnittstelle integriert, über die mittels Visual Basic auf Objekte in der Datenbasis zugegriffen werden kann und so etwa das Einlesen sinnvoll strukturierter ASCII-Dateien ermöglicht wird. Ein direkter Export nach HTML ist nicht möglich, der hersteller bietet jedoch an, XML bzw. HMTL-Reports kundenspezifisch zu erstellen.

Der Datenaustausch von Texten via Cut&Paste über die Zwischenablage wird von SELECT Enterprise anstandslos unterstützt. Grafikelemente können zwar kopiert und in entsprechenden anderen Produkten (Paint etc.) als Bilder eingefügt werden, leider besteht jedoch nicht die Möglichkeit des umgekehrten Weges des Einbinden von toolfremden Grafiken in SELECT Enterprise.

i) Code-Generierung

Code-Generierung, S. 45

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	+
8.	+
9.	+
10.	+
11.	+

Aus den im Modell enthaltenen Informationen lassen sich über verschiedene Generatoren in den entsprechenden Zielsprachen (C++, Java, u.a.) Coderahmen erzeugen. Der *C++ Generator* ist fester Bestandteil von SELECT Enterprise.

Die Codeerzeugung ist sehr flexibel an die entsprechenden Wünsche des Benutzers anpaßbar. Es können eine große Anzahl von verschiedenen Angaben für die jeweiligen Modellelemente gemacht werden. Diese umfassen etwa die automatische Erzeugung von Zugriffsfunktionen auf Attribute, die Festlegung der Behandlung von Assoziationen etc., die Deklaration von Zugriffsbeschränkungen, Defaultwerten, Argumenten und vieles mehr. Beson-

ders hervorzuheben in diesem Zusammenhang ist die Tatsache, daß die entsprechenden Einstellungen über Dialogfenster getätigt werden können. So muß der Benutzer nicht erst eine komplizierte Scriptsprache lernen, um die Codegenerierung seinen Bedürfnissen anzupassen, außerdem können für unterschiedliche Modellelemente jeweils verschiedene Angaben gemacht werden. Darüber hinaus können auch die Templates der Coderahmen verändert werden.

Teile, die außerhalb von SELECT Enterprise zum Code hinzugefügt wurden, erkennt das Tool und markiert diese als *extern*. Diese können dann in das bestehende Modell übernommen werde (siehe *Reverse-Engineering*, S. 120) und werden beim nächsten Anstoß der Codegenerierung wieder hinzugefügt.

j) Reverse-Engineering

Reverse-Engineering, S. 45

1.	*+*
2.	*+*
3.	*0*

Das Einlesen von Code und das automatische Updaten des Modells sind in SELECT Enterprise sehr effizient gelöst. Zur Anwendung kommt der sogenannte *Synchronizer* mit dessen Hilfe sowohl das Modell als auch der zugrundeliegende Programmcode konsistent gehalten werden können. Am besten läßt sich dies anhand eines Beispiels erläutern:

Im Modell *X* ist in der Datenbasis (bzw. im Klassendiagramm) eine Klasse *A* enthalten, mit zwei Attributen *a1* und *a2*, die als *private* deklariert sind. Diese können weiter spezifiziert sein, was aber hier keine nähere Erläuterung findet. Die Klasse besitzt der Einfachheit halber auch keine Operationen. Die C++ Codegenerierung wird angestoßen mit der Bedingung, daß automatisch öffentliche Zugriffsfunktionen auf die privaten Attribute erzeugt werden sollen. Dies funktioniert auch ohne Probleme.

Zusätzlich wird nach Abschluß der Codegenerierung im Klassendiagramm ein weiteres Attribut *a3* der Klasse hinzugefügt.

Nun wird der *C++ Synchronizer* aufgerufen, und das Modell wird mit den zugrundeliegenden Header-Dateien verglichen. Der Synchronizer stellt in einem geteilten Fenster jeweils graphisch die im Modell enthaltenen Elemente den im Code enthaltenen gegenüber. Es wird korrekt angezeigt, daß die entsprechenden Zugriffsfunktionen für die Attribute *a1* und *a2* zwar im Code, jedoch nicht im Modell vorhanden sind, und das Tool bietet automatisch die Möglichkeit, diese im Modell zu übernehmen. Gleichzeitig registriert der Synchronizer, daß das Attribut *a3* noch nicht im Code vorhanden ist. Mit der Auswahl der Funktion *Commit* werden Datenbasis und Code entsprechend aktualisiert, falls der Benutzer nicht anderweitige Angaben getätigt hat, etwa daß nur bestimmte Elemente "synchronisiert" werden sollen. Nach Beendigung der Funktion sind Modell und Code wieder konsistent.

Es ist auch möglich, im Synchronizer andere bestehende C++ Dateien anzugeben, die analysiert und deren Inhalt in die Datenbasis übernommen wird. Im Klassendiagramm können die Elemente (respektive Klassen, Attribute etc.) anschließend über die Funktion *Populate* automatisch graphisch angelegt werden.

k) Test / Verifikation

Test / Verifikation, S. 45

1.	*-*
2.	*+*

Zur Generierung von Testfällen bietet das Tool die Möglichkeit der Anbindung an den TestDirector der Firma Mercury Interactive's.

Eine Funktion, die SELECT Enterprise von anderen Werkzeugen unterscheidet, ist der sogenannte *Object Animator*. Hiermit kann auf Basis eines erstellten Sequenzdiagramms eine Simulation auf Modellebene durchgeführt werden. Die einzelnen Objekte werden graphisch dargestellt und das Sen-

den von Nachrichten kann durch eine gesteuerte Animation vom Anwender komfortabel verfolgt werden.

l) Maßzahlen und Statistiken

Maßzahlen und Statistiken, S. 46

1.	*0*
2.	*-*
3.	*-*
4.	*-*

Über das Zusatztool SELECT Estimator können Projektleiter anhand der im Modell enthaltenen Informationen Aufwandsschätzungen über die unterschiedlichen Entwicklungsstadien (Analyse, Design, Implementation etc.) eines Projekts durchzuführen. Zur Verwendung kommen verschiedene Metriken und andere Verfahren, SELECT Enterprise selbst stellt derartige Funktionen nicht zur Verfügung.

m) Multi-User - Eigenschaften

Multi-User - Eigenschaften, S. 46

1.	*+*
2.	*0*
3.	*+*
4.	*+*
5.	*0*
6.	*-*
7.	*-*

Durch die vollständig skalierbare und verteilte Architektur des zugrundeliegenden Repositorys Enabler wird das gleichzeitige Arbeiten mehrerer Benutzer an einem Projekt wesentlich unterstützt.

Leider ist SELECT Enterprise selbst jedoch nur auf Windows-Plattformen lauffähig (Windows 9x, Windows NT 4.0). Das Enabler Repository kann jedoch in einem Netzwerk von einem (oder mehreren, da die Datenbasis auch aufgeteilt werden kann) Windows NT-, oder UNIX Servern via TCP/IP bereitgestellt werden, als Clients können sowohl Win9x, als auch NT Rechner Anwendung finden. Die Clients nutzen einen sogenannten *Mapping* Mechanismus, um Repositorys auf Servern zu referenzieren und zu verwenden. So besteht die Möglichkeit des Zugriffs auf Projekte, die auf verschiedenen Rechnern im Intranet/Internet liegen und von diesen bereitgestellt werden. Um "off-site" zu arbeiten, kann ein Modell auch komplett auf den eigenen Rechner transferiert werden. Nach Abschluß der Arbeit, wird das Modell wieder auf dem Server gespeichert.

Enabler stellt auch sämtliche Funktionen für die kontrollierte Zusammenarbeit der Benutzer zur Verfügung, wie z.B. Locking-Mechanismen, die das unerwünschte Ändern oder Löschen eines Elements verhindern sollen. Es kommen drei Arten von Locks zur Anwendung:

- *short term application locks*, die während einer Transaktion den Zugriff von außen verhindern,
- *long-term application lock*s, die während der Ausführung einer Anwendung bestehen und
- *persistent locks*, die auch nach Beendigung der Anwendung weiterbestehen.

Um diese Funktionen für die Multi-User-Verwaltung benutzerfreundlich zu gestalten, wird bei der Installation von SELECT Enterprise das sogenannte *SELECT Enterprise Models Neighborhood* in die Windows-Umgebung integriert. Hierdurch können in einem separaten Fenster, bzw. im Windows-Explorer sämtliche Modelle durchsucht und bearbeitet werden. Zur Verfügung stehen u.a. die folgenden Möglichkeiten:

- Modelle importieren
- aktive Benutzer/Bearbeiter anzeigen
- Repositorys auf anderen Servern referenzieren
- Modelle öffnen, löschen, klonen, exportieren
- Modellversionen verwalten (Anlegen einer neuen Version des entsprechenden Modells)
- Vergabe von Zugriffsrechten

Über die Administration von Enabler können Benutzer außerdem in verschiedene Gruppen eingeteilt werden, unabhängig vom User-Konzept des Betriebssystems. Funktionen zum interaktiven Dialog zwischen den Benutzern (Nachrichten, *Talk*- Funktion) sind jedoch nicht vorhanden.

n) Security

Security, S. 46

1.	0
2.	+
3.	0
4.	+
5.	+
6.	+
7.	+
8.	0
9.	+
10.	+
11.	+

Wie bereits in *Multi-User - Eigenschaften*, S. 121 beschrieben, stellt das zugrundeliegende Repository-System Enabler sämtliche Funktionen zur konfliktfreien Zusammenarbeit von mehreren Benutzern zur Verfügung.

Es können unabhängig vom Benutzerkonzept des Betriebssystems Benutzergruppen angelegt werden, außerdem gibt es besonders ausgezeichnete Benutzer, welche die Repositorys verwalten und warten (Administratoren, Librarians). Diese legen auch die Rechte für die "normalen" Benutzer (*Model Users*) fest.

Weiters besteht die Möglichkeit, verschiedene Arten von Zugriffsrechten auf ein bestimmtes Modell jeweils auf Gruppen, oder Benutzerebene zu vergeben. Diese Zugriffsrechte umfassen

- *none* - kein Zugriff,
- *read* - nur Lesezugriff und
- *read-write* - Lese- und Schreibzugriff.

Administratoren haben die Möglichkeit, die Repositorys zu sichern (Backup), neue Repositorys anzulegen, bestehende zu löschen etc. Enabler besitzt außerdem einen effektiven Recovery-Mechanismus, der im Falle eines Systemabsturzes oder -fehlers die Wiederherstellung eines konsistenten Zustands der Datenbasis ermöglicht.

Auch für die erwähnten Administrationsaufgaben wurde eine Anwendung integriert, die sogenannte *Enabler Administration*. Unter Verwendung dieses Tools hat der Administrator Zugriff auf alle Repositorys und kann diese bearbeiten, das heißt z.B. sichern, prüfen, Einstellungen verändern, Speicherorte festlegen und weitere Parameter spezifizieren, wie die maximale Anzahl gleichzeitiger Benutzer etc.

Das Löschen von Elementen muß auf Wunsch immer bestätigt werden, es kann aber auch die Einstellung vorgenommen werden, daß eine Abfrage entfällt, falls ein Element nur aus einem Diagramm, jedoch nicht aus der Datenbasis gelöscht wird. Der zur Verfügung gestellte Undo/Redo-Mechanismus ist praktisch unbegrenzt.

o) Multi-Projekt-Eigenschaften / Versionsmanagement

Multi-Projekt-Eigenschaften / Versionsmanagement, S. 47

1.	+
2.	+
3.	+
4.	+
5.	-

Das SELECT-Paradigma der komponentenbasierten Entwicklung spiegelt sich auch beim Bearbeiten mehrerer Projekte wider, wobei bereits aus anderen Projekten vorhandene Komponenten wiederverwendet werden. Über den *SELECT Component Manager* ist ein Austausch von Elementen effizient möglich, diese können einfach in die dem eigenen Projekt zugrundeliegende Datenbasis kopiert werden. Auch können durch die Möglichkeit der Einbindung sogenannter *externer* Klassen Referenzen auf nicht im eigenen Modell enthaltener Elemente gelegt werden.

SELECT Enterprise bietet auch die Möglichkeit, unterschiedliche Versionen eines Projekts zu verwalten. Wie bereits in *Multi-User - Eigenschaften*, S. 121 erwähnt, kann mit Hilfe des *SELECT Enterprise Models Neighborhood* eine neue Version des aktuellen Modells angelegt werden, mit der defaultmäßig dann weitergearbeitet werden kann. Der Zugriff auf die alten Versionen ist ebenfalls jederzeit möglich.

5.6.3 Ergonomie

a) Benutzeroberfläche

Benutzeroberfläche, S. 47

1.	+
2.	+
3.	+
4.	+
5.	+
6.	*0*
7.	+

Das graphische Userinterface von SELECT Enterprise wurde von den Entwicklern mit großer Sorgfalt an die Standardoberfläche von Windows angepaßt. Es enthält sämtliche Funktionen, die von einem modernen Windows-Programm erwartet werden: *Multiple Document Interface*; verschiedene Teilbereiche des Hauptfensters (Explorer-, Diagramm-, Ausgabe- und Eigenschaftenfenster), deren Größe anderbar ist; Standardschaltflächen und -menüs, wobei derzeit nicht verfügbare ausgeblendet sind etc.

Im Explorer-Fenster können vier verschiedene, jeweils baumstrukturartig aufgebaute Inhalte angezeigt werden, zwischen denen mittels Karteikartenreitern gewechselt werden kann:

- *Diagrams* (sämtliche Diagrammtypen)
- *Dictionary* (Inhalt der Datenbasis)
- *Relationships* (Beziehungen im statischen, sowie dynamischen Modell)
- *Help* (Online-Hilfe)

Der Repository-Browser ist sowohl im Aussehen, wie auch in der Funktionalität dem Windows-Explorer nachempfunden, was die Arbeit sehr erleichtert.

Darüber hinaus wurde auch die Funktion implementiert, Menüs und verschiedene Fenster (alle, bis auf das Diagrammfenster) zu verschieben und auf Wunsch zu verankern. Je nach geöffnetem Diagrammtyp werden unterschiedliche Einträge bzw. Symbolleisten angezeigt. Leider können die Menüs (bis auf den Eintrag *Tools*, über den externe Werkzeuge wie Code-Generator u.a. gestartet werden), sowie die Symbolleisten nicht mit neuen Einträgen verändert werden.

Besonderer Wert wurde auch auf die Ausnutzung des Kontextmenüs gelegt, das wie gewohnt über die rechte Maustaste aufrufbar ist. Gleich welches Element (ob ein Element in einem Diagramm, oder ein Eintrag im Data-Dictionary), über das Kontextmenü lassen sich die entsprechend relevanten Aktionen starten (z.B. Anzeigen des Zustandsdiagramms zu einer Klasse).

Leider hat die Implementation der graphischen Benutzeroberfläche von SELECT Enterprise auch Nachteile. Es ist fast unmöglich, auf einem 17-Zoll Monitor in der Auflösung 1024*768 (welches durchaus als Standard angesehen werden kann) die Übersicht über das gerade bearbeitete Diagramm zu behalten, ohne nicht die übrigen Teile des Arbeitsbereichs (Repository-Explorer, Ausgabe-, sowie Eigenschaftenfenster) mehr oder minder auszublenden. So konnte das Klassendiagramm des in Kapitel 6 bearbeiteten Beispielprojekts nur komplett dargestellt werden, wenn der Zoomfaktor auf 37% eingestellt war. Natürlich ist eine derartige Verkleinerung kaum mehr effizient. Verkleinert der Benutzer jedoch die übrigen angesprochenen Fenster, so fehlt natürlich die Möglichkeit der effektiven Informationsbeschaffung (Browsen im Repository, Anzeigen von Ergebnissen sowie Ändern der Eigenschaften eines Elements). So ist der Benutzer leider gezwungen, im Arbeitsprozeß regelmäßig Kompromisse zwischen Ansicht und Information eingehen. Als "Trick" bleibt etwa, das Ausgabe- und das Eigenschaftenfenster nicht zu verankern und zu schließen. Wird dann im Kontextmenü eines Elements z.B. der Eintrag *Properties* gewählt, so wird das Fenster im Arbeitsbereich eingeblendet, jedoch muß es auch wieder manuell geschlossen werden. Hier besteht noch etwas Änderungsbedarf, da man nicht verlangen kann, daß alle Benutzer Monitore mit einer Größe von mindestens 21-Zoll verfügen.

b) Tool-Dokumentation

Die Dokumentation zu SELECT Enterprise setzt vor allem auf die vorhandene Online-Hilfe und den *Process-Mentor* (siehe *Hilfefunktionen*, S. 125). Sämtliche Dokumente liegen nur in englischer Sprache vor.

Mitgeliefert werden u.a. ein Handbuch zur Installation von SELECT Enterprise und zur Administration des Repositorys Enabler, die auch als PDF-Dateien auf der Installations-CD vorliegen, wie auch eine Übersicht über Änderungen und Erweiterungen der aktuellen Werkzeugversion.

Zwei umfangreiche Beispielprojekte liegen dem Programmpaket ebenfalls bei.

Tool-Dokumentation, S. 48

1.	0
2.	+
3.	-
4.	-
5.	0
6.	+
7.	0
8.	+
9.	+
10.	+
11.	+
12.	+

c) Usability

Wie bereits in *Benutzeroberfläche*, S. 123 beschrieben, haben die Vorteile der Benutzeroberfläche von SELECT Enterprise auch ihre Schattenseiten. Besonders der offensichtliche Versuch der Entwickler, Dialoge und Fenster möglichst aufgeräumt und übersichtlich zu gestalten, ist in bestimmten Bereichen eher hinderlich für einen flüssigen Arbeitsfortschritt. Dies soll wieder anhand eines kleinen Beispiels erläutert werden:

Angenommen, der Benutzer legt eine neue Klasse *A* an. Der Bezeichner der Klasse kann sofort im Diagrammfenster vergeben werden. Sollen jedoch weitere Angaben über die Klasse in das Modell eingetragen werden (z.B. der Typ *abstrakt*), so muß auf das entsprechende Property-Fenster zurückgegriffen werden. Jetzt sollen mehrere Attribute der Klasse festgelegt werden. Der Benutzer legt diese nacheinander zusammen mit ihren Bezeichnern im Klassendiagramm an. Um die Eigenschaften der Attribute zu bearbeiten (Datentyp etc.), muß nacheinander jedes Attribut ausgewählt und die Angaben im Property-Fenster getätigt werden.

Am ausgeprägtesten zu beobachten ist dieses Manko anhand des Anlegens von Methoden / Operationen einer Klasse. Wie vorher werden die einzelnen Operationen und deren Bezeichner im Klassendiagramm festgelegt. Die genauere Spezifizierung der Eigenschaften führt den Benutzer für jede Operation eigens durch eine Anzahl von Karteikarten des Property-Fensters (siehe Abb. 5–1):

Usability, S. 48

1.	0
2.	0
3.	+
4.	0
5.	+
6.	+
7.	+
8.	0
9.	+
10.	+
11.	-

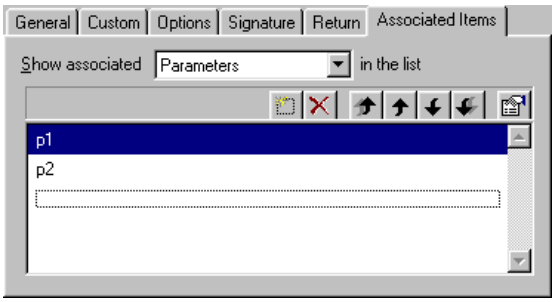

Abb. 5–1
Das "Property"-Fenster von SELECT Enterprise

- *Options* - Festlegung von Kriterien wie Sichtbarkeit u.ä.,
- *Return* - Festlegung des Rückgabewert-Typs und schließlich
- *Associated Items* - Festlegung der Parameterliste.

In der zuletzt erwähnten Karteikarte kann wiederum nur der Bezeichner des jeweiligen Parameters eingegeben werden, um den Typ desselben festzulegen, muß ein weiterer *Property*-Dialog geöffnet werden, usw.

Hieran ist unschwer zu erkennen, daß in diesem Bereich (und dieses Beispiel ist eine Grundfunktion beim Anlegen eines Klassendiagramms) noch

große Verbesserungen notwendig sind. Eine Möglichkeit wären etwa die in anderen Tools vorhandenen Tabellen-Editoren, mit deren Hilfe sämtliche Inhalte einer Klasse zusammen mit ihren Eigenschaften tabellarisch aufgelistet und innerhalb eines Dialogfensters geändert werden können.

Die übrigen Funktionen, die von einer Windows-Anwendung erwartet werde, wie Tastaturshortcuts, Symbolleisten, Statuszeile und Fortschrittsbalken sind effizient implementiert.

SELECT Enterprise wird nur in englischer Sprache ausgeliefert.

d) Hilfefunktionen

Hilfefunktionen, S. 49

1.	+
2.	+
3.	+
4.	+

Das Hilfesystem von SELECT Enterprise ist sehr gut und umfangreich und bietet neben der klassischen Hilfefunktionen, wie Inhalt, Index und Suchfunktion, auch die Möglichkeit mittels *Answer Work* eigene natürlichsprachliche Fragen zu formulieren, die interpretiert werden.

Zu erreichen ist die Hilfe entweder über den entsprechenden Menüeintrag oder über die Taste *F1*, weiters kann auch das Browserfenster zum Durchsuchen der Inhalte benutzt werden. Weitere Funktionen, wie Pop-Up Hilfe beim Verweilen über einem Symbol, und kontextsensitive Hilfe unterstützen den Benutzer bei der Arbeit mit SELECT Enterprise. Die bereitgestellten Informationen beschreiben die entsprechenden Bereiche ausführlich und gut verständlich.

Abschließend ist noch auf den sogenannten *Process Mentor* hinzuweisen, der in ansprechende Weise allgemeine und praxisbezogene Informationen zum Projektablauf (Phasen, UML, Modellierungstips etc.) zur Verfügung stellt.

e) Graphikeditor / Darstellung

Graphikeditor / Darstellung, S. 49

1.	*0*
2.	+
3.	+
4.	+
5.	+
6.	+
7.	*0*
8.	+
9.	+
10.	*0*
11.	*0*
12.	-
13.	+
14.	-
15.	+

Die Darstellung der Diagrammelemente ist ansprechend, es können auch Voreinstellungen zur unterschiedlichen Farbgestaltung bestimmter Elemente (Klassen, Use-Cases, u.a.) getätigt werden. Der Ausdruck eines Diagramms entspricht genau dem des Bildschirminhalts.

Über den Typ des *General Graphics Diagram* können Diagramme mit vordefinierten, jedoch nicht in der UML enthaltenen Elementen (Ovale, Rechtecke etc.) erstellt werden, in Klassendiagrammen u.a. besteht diese Möglichkeit nicht.

SELECT Enterprise stellt im Diagrammeditor auch verschiedene Alignment-Funktionen zur Verfügung, mit deren Hilfe die Elemente (alle, oder nur ausgewählte) untereinander ausgerichtet werden. Seltsamerweise können Klassen eines Package nicht ausgerichtet werden.

Die Größe von Elementen kann lediglich für bestimmte Typen (z.B. Pakkages) geändert werden, Klassen etwa werden in ihrer Größe immer automatisch angepaßt, eine Voreinstellung fehlt ebenso, wie Textumbruch innerhalb eines Elements. Außerdem gibt es keine Möglichkeit zur Einstellung der Schriftgröße oder Strichstärke.

Linien von Assoziationen, Generalisationen u.a. können entweder gerade verlaufen, oder mittels sogenannter *Waypoints* um beliebige andere Elemente herum geführt werden. Diese Waypoints lassen sich nachträglich auch einfach über den Kontextmenüeintrag einer Beziehung wieder entfernen. Besonders hervorzuheben ist auch der Umstand, daß Linien im Prinzip immer optimal geführt werden, es gibt keine Überschneidungen und falls doch, so kann über die zuvor beschriebene Funktion eine ansprechende Darstellung erzeugt werden.

Wird ein Modell erneut geladen, so entspricht die Darstellung der Diagramme wieder der zuletzt bearbeiteten.

f) Graphikeditor / Bedienung

Graphikeditor / Bedienung, S. 50

Die Notationselemente werden aus einer Symbolleiste gewählt und durch einfacher Mausklick auf der Zeichenfläche abgelegt. Dort können sie per Drag&Drop verschoben werden. Anschließend fordert SELECT Enterprise den Benutzer auf, einen Bezeichner für das angelegte Element einzugeben, Elemente ohne Bezeichner können nicht existieren. Das Abbrechen einer ungewollten Aktion ist möglich.

Der Zeichenbereich ist nicht in der Größe eingeschränkt, es gibt die gewohnten Scrolling-Möglichkeiten. Zur Mehrfachauswahl bestimmter Elemente können diese über eine *Lasso*-Funktion markiert werden. Anschießend kann auf alle Elemente innerhalb der Auswahl eine Aktion durchgeführt werden.

Auch eine Grid-Alignment Funktion zum Ausrichten der Elemente anhand eines Gitternetzes ist vorhanden. Dieses kann auf Wunsch angezeigt oder ausgeblendet werden. Es gibt eine Undo wie auch Redo-Funktion, die in unbegrenzter Tiefe auch Aktionen wie das Verschieben von Elementen rückgängig machen können.

Als weitere gängige Windows-Funktionen wurden Kopieren, Ausschneiden, Einfügen und das Löschen mittels der Taste *Entf* implementiert.

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	+
8.	+
9.	+
10.	+
11.	0
12.	+
13.	+
14.	+

g) Performance

Performance, S. 51

Es ist mit SELECT Enterprise ein durchaus zügiges Arbeiten möglich, vor allem durch die Windows-basierte Benutzeroberfläche erfolgt der Diagrammaufbau und die Manipulation der Inhalte relativ effizient, mit Abstrichen bei der zu geringen Größe des Diagrammfensters (siehe *Benutzeroberfläche*, S. 123).

Das Laden von gesamten Projekten beansprucht natürlich eine gewisse Zeit, jedoch liegt diese meist im Sekundenbereich. Da das Updaten des Repository automatisch im Hintergrund durchgeführt wird, entfällt ein Speichervorgang vollkommen.

Vor dem Starten einer komplexen Suchabfrage innerhalb der kompletten Datenbasis wird der Benutzer darauf hingewiesen, daß dieser Vorgang eventuell einen größeren Zeitraum beanspruchen könnte.

Die Codegenerierung und das Einlesen von bestehendem Code wird ebenfalls in ausreichendem Tempo ausgeführt.

1.	+
2.	+
3.	+
4.	0

h) Adaptivität

Adaptivität, S. 51

SELECT Enterprise stellt keine Funktionen zur Verfügung, entsprechend des Lernfortschritts des Anwenders die Menü- und Symbolleisteneinträge zu verändern und in ihrem Umfang anzupassen. Wie bereits erwähnt, können nur zum *Tool*-Menü eigene Einträge hinzugefügt werden, um externe Werkzeuge starten zu können.

Der Soucecode von SELECT Enterprise wird nicht mitgeliefert.

1.	-
2.	-
3.	0
4.	-
5.	+
6.	-

5.7 Software through Pictures/UML 7.1

Software through Pictures/UML wurde in der Version 7.x von einem UNIX-Programm nach Windows portiert. Die Komplexität und Leistungsfähigkeit des Werkzeugs prädestinieren es rein für den Einsatz in großen Projekten. Die Datenhaltung wird entweder über das relationale DBMS Sybase SQL Server, oder über das Microsoft Jet Repository abgewickelt, welches zwar eine geringere Funktionalität, jedoch auch einen kleineren Ressourcenbedarf besitzt.

StP/UML ist nicht wirklich intuitiv zu bedienen, so fehlt etwa ein herkömmlicher Repository-Browser, auch der übrige Teil des Werkzeugs wirkt relativ spartanisch. Ein Großteil der Einstellungen wird nur über Scripts geregelt, die vom Benutzer den persönlichen Anforderungen gemäß angepaßt werden können.

5.7.1 Methodenbasierte Kriterien

a) Modellierung der Objektstruktur - Objektstatik

Modellierung der Objektstruktur - Objektstatik, S. 38

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	*0*

Elemente, die der Benutzer in einem Diagrammtyp einzeichnet, werden automatisch im zugrundeliegenden Data-Dictionary des aktuellen Projekts gespeichert.

Ein bestimmtes Element (z.B. eine Klasse) kann in mehreren Diagrammen (z.B. Klassendiagrammen) bzw. auch in einem Diagramm mehrmals aufscheinen. Daher wird auch durch die Konsistenzprüfung etwa ein Anlegen von zwei Klassen mit demselben Bezeichner nicht kritisiert. Beide Diagrammelemente beziehen sich auf dieselbe Klasse, eine Änderung im einen Element zieht eine Änderung im anderen mit sich. Die Konsistenzprüfung muß im übrigen manuell angestoßen werden und läuft nicht automatisiert im Hintergrund ab.

Die Informationen können nachträglich verändert werden (z.B. Bezeichner einer Klasse, Typ eines Attributs etc.), die Änderung erfolgt im Allgemeinen für alle Repräsentationen des entsprechenden Elements in verschiedenen Diagrammen. Dies wird ebenfalls durch die Funktion *Rename Objects Systemwide* unterstützt, mittels derer der Bezeichner einer Klasse global in der Datenbasis, d.h. für sämtliche Repräsentationen dieser Klasse geändert werden kann.

Die Komponenten einer Klasse werden normalerweise alphabetisch geordnet, die Reihenfolge kann jedoch vom Benutzer auch geändert werden.

Im Klassendiagramm selbst können für Attribute und Operationen nur deren Bezeichner eingetragen werden, alle weiteren Informationen werden in den sogenannten *Class Tables* festgelegt. Dazu gehören:

- Sichtbarkeit von Klassenelementen
- Typen und Defaultwerte von Attributen
- Parameterliste und Rückgabewerte von Operationen
- Art der Übergabeparameter (*in*, *out*, *in/out*)
- *constant*-Vermerk (C++)
- *inline*-Anweisung (C++)
- weitere sprachspezifische Informationen

Die hierfür verwendeten *Table-Editoren* repräsentieren ein durchgängiges Konzept in StP, um Diagramm- von Detailinformationen zu trennen. Die Informationen zu einem Element der Datenbasis werden tabellarisch ange-

zeigt und können komfortabel geändert werden. Dies ist vor allem für den Locking-Mechanismus im Multi-User Betrieb notwendig, da ja etwa wie erwähnt, ein und dieselbe Klasse in mehreren Klassendiagrammen verwendet werden kann. Würden nun verschiedene Benutzer während der Arbeit an unterschiedlichen Diagrammen die Klassenstruktur ändern, so könnte diese nur schwer konsistent gehalten werden. So setzt jedoch ein Benutzer mit Öffnen der Klassentabelle einen Lock auf das entsprechende Element und kann die Änderungen durchführen, es ist jedoch für die übrigen Benutzer weiter möglich, die Klasse zu verwenden.

b) Modellierung der Systemstruktur - Systemstatik

Modellierung der Systemstruktur - Systemstatik, S. 39

1.	*0*
2.	*+*
3.	*0*
4.	*0*

Rekursive Aggregationsbeziehungen können angelegt werden. Leider ist deren Darstellung nicht ganz UML-konform, da etwa an beiden Enden der Aggregation Diamantsymbole angezeigt werden.

Einmal erfaßte Beziehungen können nachträglich mittel Drag&Drop verändert werden, die Informationen im Eigenschaftendialog (Bezugsklassen) werden automatisch aktualisiert.

Die Einteilung von Klassen erfolgt nur UML-konform mittels Packages, bzw. mit Hilfe von Stereotypen, eine weitere toolspezifische Einteilung ist nicht vorgesehen.

OCL (*Object Constraint Language*)-Annotationen werden nicht direkt unterstützt, sondern müssen vom Benutzer als Freitext eingegeben werden und werden von StP weder interpretiert noch auf ihre Syntax hin überprüft.

c) Dynamisches Objektverhalten - Objektdynamik

Dynamisches Objektverhalten - Objektdynamik, S. 39

1.	*+*
2.	*0*
3.	*0*
4.	*0*
5.	*+*
6.	*-*

Zustandsdiagramme sind eindeutig mit einer Klasse verbunden. Hierzu wird die entsprechende Klasse markiert und im Menü *GoTo* der Punkt *State Diagram for Class* ausgewählt wird. Ist noch kein Zustandsdiagramm vorhanden, wird abgefragt, ob ein neues erstellt werden soll.

Zustände können im Diagramm mit einem Bezeichner versehen werden, eine genauere Definition der Eigenschaften (Aktivitäten, Ereignisse) wird wieder über einen Table-Editor abgewickelt. Es können unter anderem folgende Eigenschaften angegeben werden:

- ❑ Aktion beim Zustandeintritt (*entry-action*)
- ❑ andauernde Aktivität (*do activity*)
- ❑ Aktion beim Verlassen des Zustands (*exit action*)
- ❑ interne Ereignisse und Aktionen (*internal events / actions*)
- ❑ Zustandsvariablen
- ❑ aufgeschobene Ereignisse (*deferred events*)

Leider ist es nur unzureichend möglich, Zustandsübergänge genauer zu beschreiben. Dies erfolgt lediglich über textuelle Beschreibung des Bezeichners der entsprechenden Transition. Dadurch entfällt auch eine Syntaxprüfung der Angaben.

Zustände können sehr komfortabel verfeinert (*generalisiert*) werden. Der State-Diagram Editor von StP kann sowohl UND- (*concurrent states*) als auch ODER- (*sequential states*) verfeinerte Zustände korrekt geschachtelt darstellen. Hierzu bestehen zwei Möglichkeiten: Entweder es wird der Superzustand mit den darin enthaltenen Subzuständen direkt im aktuellen Diagramm eingezeichnet, oder der zu verfeinernde Superzustand wird markiert und im Menü *GoTo* der Punkt *Refine* angewählt. Ein neues Diagramm wird angelegt, in dem die entsprechenden Elemente angelegt werden können. Im

übergeordneten Diagramm wird der Superzustand UML-konform als verfeinert dargestellt. Ereignisse selbst können nicht generalisiert werden.

d) Dynamisches Systemverhalten - Systemdynamik

Dynamisches Systemverhalten - Systemdynamik, S. 39

1.	0
2.	+

Sequenzdiagramme können erstellt werden, indem ein entsprechender Anwendungsfall (Use-Case) ausgewählt und über einen Menübefehl eine neues Sequenzdiagramm angelegt wird. So ist auch die Kopplung zwischen Use-Case und Sequenzdiagramm sichergestellt.

Der Editor von StP bietet im Prinzip alle von der UML geforderten Elemente eines Sequenzdiagramms, so können neben Lebenslinien eines Objekts auch deren Erstellung und Löschung eingezeichnet werden. Die Zuordnung eines Objekts zu einer bestimmten Klasse erfolgt etwas uneffizient über den Bezeichner des Objekts, hierbei muß der Objektname und davon mittels Doppelpunkt getrennt der Klassenname eingetragen werden. Eine direkte Auswahl der im Projekt vorhandenen Klassen ist nicht möglich.

Zur Spezifikation einer Nachricht können Bezeichner von vorhandenen Methoden einer Klasse ausgewählt werden. Es besteht unter anderem auch die Möglichkeit, für Nachrichten aktuelle Parameter- und Rückgabewerte, sowie Überwachungsbedingungen anzugeben. Auf Attribute einer Klasse kann jedoch nicht Bezug genommen werden.

e) Zusammenhang zwischen den Modellen

Zusammenhang zwischen den Modellen, S. 39

1.	+
2.	-
3.	+
4.	-

Ein Zustandsdiagramm ist umkehrbar eindeutig mit einer Klasse verbunden (siehe *Dynamisches Objektverhalten - Objektdynamik*, S. 128).

f) Abstraktionsebenen / Sichten

Abstraktionsebenen / Sichten, S. 40

1.	+
2.	+
3.	0
4.	0
5.	+

Der Detaillierungsgrad der Anzeige läßt sich komfortabel einstellen. Dies erfolgt mit Hilfe von Filtern, die für eine bestimmtes Diagramm eingerichtet werden. Diese sind im Prinzip StP-spezifische Scripts, die so auch den Bedürfnissen des Anwenders entsprechend angepaßt werden können. Bereits im Lieferumfang enthalten sind unter anderem Scripts zur Anzeige aller Elemente (*Show All*), bzw. zum Verstecken von bestimmten Elementen (*Hide Links*, *Hide Unselected Symbols*, *Hide Comments*, u.a.). Parameterlisten von Operationen sowie Eigenschaften der Attribute einer Klasse lassen sich so ebenfalls auf Wunsch anzeigen, oder nicht. Die Einstellung bezieht sich auf Wunsch auf das entsprechende Diagramm, oder auf das gesamte Projekt.

Verfeinerte Zustandsdiagramme lassen sich nur entweder gesamt (Super- und Subzustände) anzeigen, oder nur der Superzustand wird dargestellt, falls dieser über die in *Dynamisches Objektverhalten - Objektdynamik*, S. 128 beschriebene Funktion verfeinert wurde.

g) Vorgangsreihenfolge bei der Entwicklung

Vorgangsreihenfolge bei der Entwicklung, S. 40

1.	+
2.	0
3.	+
4.	-

StP zwingt den Benutzer nicht zu einer bestimmten Vorgehensweise bei der Erstellung des Projekts. Es können Elemente lediglich zusammen mit ihrem Bezeichner angelegt und dann später genauer spezifiziert werden. Das Anlegen der Elemente erfolgt jedoch im Prinzip nur innerhalb eines Diagramms, die Informationen können nur über Umwege direkt in das Data-Dictionary eingetragen werden.

Dies ist auch daran zu erkennen, daß StP im Gegensatz zu anderen Tools keinen eigenen Repository-Browser zur Verfügung stellt. Das Durchsuchen der Datenbasis erfolgt nur mittels Abfragen (ähnlich Datenbank-Queries),

die Ergebnisse werden wieder in einem Tabelleneditor dargestellt. Dies ist ein großes Manko von StP, welches die intuitive Benutzung des Tools (besonders in Hinsicht auf die Möglichkeiten einer modernen graphischen Benutzeroberfläche) sehr einschränkt und dem Anwender einen großen Einarbeitungsaufwand abverlangt.

h) Unterstützung bei der Methodenanwendung

Unterstützung bei der Methodenanwendung, S. 40

1.	*0*
2.	*+*
3.	*+*
4.	*+*
5.	*0*
6.	*+*

Die Diagrammelemente entsprechen weitgehend denen des UML 1.1 Standards, so daß entsprechende Basisliteratur ohne Probleme Verwendung finden kann. Eine online abrufbare Erklärung der Notation sowie der Semantik der UML ist nicht vorgesehen, jedoch kann der Benutzer auf Wunsch auf die umfangreiche Dokumentation zurückgreifen, die auch als PDF-Dateien zur Verfügung steht und aus dem Tool aufrufbar ist. Hierzu wird für genauere Details auf *Tool-Dokumentation*, S. 139 verwiesen.

Bei Verletzungen der Konsistenz werden lediglich die entsprechenden Fehlermeldungen ausgegeben und es besteht die Möglichkeit, das Element eines Diagramms anzeigen zu lassen, das den Fehler verursacht hat. Eine weiterführende Erläuterung bleibt jedoch aus.

5.7.2 Funktionale Kriterien

a) Datenhaltung

Datenhaltung, S. 41

1.	*+*
2.	*+*
3.	*+*
4.	*0*
5.	*+*
6.	*-*
7.	*+*
8.	*+*

StP benutzt als Datenbasis zur Speicherung der in einem (oder mehreren) Projekten enthaltenen Informationen eine relationale Datenbank der Firma Sybase (in der aktuellen Version den Sybase System 11 SQL Server). Dieses ist ein sehr angesehenes und leistungsfähiges Produkt, womit eine effiziente Verwaltung der Daten gewährleistet ist, vor allem in Hinsicht auf Erweiterbarkeit, Offenheit (Schnittstellen) und die Unterstützung des verteilten und geregelten Zugriffs einer großen Anzahl von Benutzern/Entwicklern das gemeinsame Data-Dictionary (siehe *Multi-User - Eigenschaften*, S. 137).

Seit der StP Version 7.0 (die erstmals eine reine Windows-NT Unterstützung besitzt), verfügt das Tool auch über eine standardisierte ODBC-Schnittstelle, wodurch der Benutzer nicht mehr nur auf die Sybase Datenbank limitiert ist. Es steht bei der Installation auch das Microsoft Jet Repository als Datenbasis zur Auswahl. Dies ist vorwiegend deshalb von Vorteil, da Sybase besonders große Anforderungen an die Ressourcen des verwendeten Rechners stellt (z.B. Hauptspeicher) und daher für kleinere Projekte eher nicht geeignet ist.

Die Integrität der Daten und die Vermeidung von Redundanz mehrfach vorkommender Informationen wird durch das Datenbanksystem selbst garantiert. Der Modellierungsteil des Tools ist hiervon völlig unabhängig, was auch an der allgemeinen Architektur von StP abgelesen werden kann.

So können auch Informationen zwar im Repository vorkommen, jedoch nicht in einem Diagram eingezeichnet sein. Außerdem ist es so möglich, daß etwa ein und dieselbe Klasse in mehreren Diagrammen verwendet wird, die Informationen jedoch konsistent gehalten werden (siehe *Modellierung der Objektstruktur - Objektstatik*, S. 127).

StP bietet darüber hinaus die Möglichkeit, Daten auch direkt in das Repository einzugeben. Hierfür stehen einerseits die verschiedenen Table Editoren, auf der anderen Seite der *Repository Browser* selbst zur Verfügung (siehe *Browsing-Eigenschaften*, S. 131).

StP ist folgendermaßen aufgebaut: Den Grundstock, bzw. den Kern von StP bildet der sogenannte *StP Core*. In diesem sind unter anderem die Datenbasis und das Meta-Modell enthalten. Erst darauf aufbauend finden sich die eigentlichen Modellierungswerkzeuge für die unterschiedlichen Methoden, wie UML, OMT, Booch, oder auch StP/IM zu Modellierung von relationalen Datenbanken (ER-Diagramme), sowie verschiedene andere Werkzeuge (z.B. Validator/Req für Anforderungsanalyse und Test). Diese Struktur ermöglicht eine effiziente und sichere Trennung von Daten und Modellierungsinformationen eines Projekts.

StP bietet eine definierte Schnittstelle (*stpem*), über die Daten in das System eingegeben werden können. Es ist sowohl möglich, Informationen in Tabellen, als auch Diagramminformationen direkt von außen einzutragen.

b) Browsing-Eigenschaften

Browsing-Eigenschaften, S. 41

1.	0
2.	+
3.	-
4.	+
5.	-
6.	0
7.	0
8.	0
9.	-
10.	+
11.	0

Das Suchen nach Elementen in einem StP-Projekt erfolgt einerseits über den Repository Browser, es werden jedoch auch Funktionen etwa zum Auffinden einer bestimmten Klasse in einem Diagramm geboten.

Der Repository Browser ist im Gegensatz zu anderen Tools nicht Windows-konform als Baumstruktur ausgeführt sondern präsentiert sich im Prinzip als große Tabelle (ähnlich dem Erscheinungsbild der bereits angesprochenen Tabelleneditoren), in der die gewünschten Informationen der Datenbasis dargestellt werden. Leider ist dadurch die Bedienung desselben für den Anwender nicht so intuitiv wie bei anderen Werkzeugen. Um das Repository zu durchsuchen, muß der Benutzer Datenbankabfragen in der Scriptsprache *OMS* (*Object Management Statements*) formulieren, die Ergebnisse werden wiederum tabellarisch aufgelistet. Die Funktionalität solcher Abfragen ist umfassend - es können sowohl Diagramminhalte, als auch auf andere projektspezifische Daten (wie Benutzerrechte, zugeordnete Dokumente etc.) durchsucht und angezeigt werden. Leider ist es nicht möglich, von den Ergebnisse der Abfrage direkt in das entsprechende Diagramm überzuwechseln, z.B. kann beim Auffinden einer bestimmten Klasse nicht in das entsprechende Klassendiagramm gewechselt werden, in der diese verwendet wird.

Die Syntax von OMS soll anhand eines Beispiels dargestellt werden. Die folgende Suchabfrage listet alle Operationen (alphabetisch nach Bezeichnern sortiert) eines Projekts auf, in denen der Substring *msg* vorkommt.

```
node [type=UMLOperation && name $ '*msg*']
sort by name
```

Es ist offensichtlich, daß eine direkte Datenbankabfrage über Scripts sehr mächtig und flexibel ist. Jedoch fehlt (wie bereits erwähnt) die Intuitivität der Darstellung der Informationen und die einfache Bedienbarkeit, die ein Browser (der in diesem Fall im herkömmlichen Sinne eigentlich keiner ist) bieten sollte. Da im Hauptfenster von StP ja eine solche Möglichkeit zum Durchsuchen etwa der verschiedenen Diagramme bereits implementiert ist, stellt sich die Frage, warum diese Struktur nicht auch für die Datenbasis zu Verfügung steht. Zusätzlich ist Performanz der Abfragen eher gering, was den Arbeitsprozeß zusätzlich behindert.

c) Navigation im Systemmodell

Die Suche nach bestimmten Elementen des Projekts wird durch die in *Browsing-Eigenschaften*, S. 131 beschriebenen Möglichkeiten der Datenbankab-

frage durchgeführt. Hierbei werden auch Funktionen wie Pattern Matching oder gefilterte Suche effizient unterstützt.

Die Navigation im Modell selbst wird durch den Menüpunkt *GoTo* realisiert. Ist etwa ein Zustandsdiagramm einer bestimmten Klasse zugeordnet worden, so kann über den Menüeintrag *State Diagram for Class* zum entsprechenden Diagramm navigiert werden.

Navigation im Systemmodell, S. 42

1.	0
2.	+
3.	0
4.	+

Weiters stellt StP eine komfortable Funktion zur Suche nach, in einem bestimmten Diagramm enthaltenen Elementen (Klassen, Assoziationen etc.) zur Verfügung. Hierfür wird ein Dialogfenster verwendet, in dem der Bezeichner des Objekts eingegeben wird. Zusätzliche Einschränkungen umfassen u.a.:

- ❑ Suche nach Symbolen und/oder Beziehungen
- ❑ Unterscheidung von Groß-/Kleinschreibung
- ❑ Legen des Ansichtsfokus auf das gefundene Element nach Abschluß der Suche

Elementspezifische Informationen lassen sich im allgemeinen über das Kontextmenü (bzw. Punkt *Properties* im Kontextmenü) abfragen.

d) Konsistenzprüfung / Integrität

Eine "Online"-Konsistenzprüfung der Diagramminformationen ist in StP nicht implementiert, so ist es möglich, während der Arbeit mit dem Tool unkorrekte Eingaben zu tätigen.

Konsistenzprüfung / Integrität, S. 42

1.	-
2.	+
3.	+
4.	0
5.	-
6.	0
7.	+
8.	0
9.	+

Der Benutzer muß eine Prüfung des Modells explizit anstoßen. Diese wird wiederum über Scripts geregelt, welche den spezifischen Anforderungen genau angepaßt werden können. So ist es dann auch möglich, z.B. durch Komplettierung des Bezeichners in Abhängigkeit vom Anfangsbuchstaben, oder der Auswahl des Bezeichners aus einer Liste von Möglichkeiten, der Vergabe von fehlerhaften oder ähnlichen Namen vorzubeugen.

Bestimmte andere Eingaben, die gegen die Konsistenz und Richtigkeit des Modells verstoßen würden, werden jedoch von StP im Vorhinein verhindert. Hierzu gehören u.a. Beziehungen, die zwar einen Start-, jedoch kein Endobjekt besitzen, oder Elemente ohne Bezeichner. Leider ist es jedoch möglich, mehrere Elemente mit gleichem Bezeichner in den Diagrammen einzutragen. So können etwa in einem Klassendiagramm zwei Klassen mit dem selben Namen aufscheinen, was auch noch aus Gründen der Übersicht sinnvoll sein könnte. Jedoch kann der Benutzer die Inhalte der Klassen unterschiedlich gestalten, etwa so, daß die eine Klasse im Diagramm bestimmte Attribute enthält und die zweite Klasse andere. Dies wird auch von der Konsistenzprüfung nicht kritisiert, sofern diese nicht anderweitig angepaßt worden ist. In der Datenbasis beziehen sich die unterschiedlichen Darstellungen dann jedoch auf ein und dieselbe Klasse, und erst nach einem Update-Prozeß über die Tabelleneditoren werden die Diagrammelemente korrekt dargestellt.

e) Report-Ausgabe

Die Generierung von Reports wird von StP wiederum über Scripts (in der Scriptsprache QRL) geregelt. Diese können vom Benutzer ohne große Probleme über eine Edit-Funktion der verschiedenen Scriptvariablen an die speziellen Anforderungen angepaßt werden.

Dem Programmpaket liegen bereits verschiedene fertige *UML-Reports* bei und zwar in den Sprachen Deutsch, Englisch und Französisch. Diese sind in der Lage, Dokumente in den folgenden Formaten zu erzeugen:

- ❑ RTF (MS-Word)
- ❑ HTML
- ❑ FrameMaker
- ❑ ASCII

Report-Ausgabe, S. 43

1.	*0*
2.	*+*
3.	*0*
4.	*0*
5.	*0*
6.	*+*
7.	*-*
8.	*0*
9.	*+*
10.	*+*

Die generierten Zusammenfassungen der aktuellen Projektdaten wirken umfangreich und gut strukturiert. Der Report bietet etwa eine gute Übersicht über folgende Inhalte (zusammen mit genaueren Beschreibungen der Modell-Details, wie Anmerkungen, Bearbeiter etc.):

- ❑ Anforderungen
- ❑ Anwendersicht
- ❑ Systemarchitektur
- ❑ Paketübersicht
- ❑ Klassenübersicht
- ❑ Stereotypen

Leider werden die Abbildungen der einzelnen Diagramme nicht wie erwartet in einem Bildformat (wie GIF oder JPG) gespeichert, sondern liegen standardmäßig eingebettet im RTF-Format vor, was den direkten Web-Export ein wenig erschwert.

f) Datenbankschema-Ausgabe

Datenbankschema-Ausgabe, S. 43

1.	*0*
2.	*-*
3.	*-*
4.	*0*
5.	*-*
6.	*0*

Die Modellierung von relationalen Datenbanken aus dem statischen Modell heraus ist in StP/UML direkt nicht möglich. Hierzu ist es notwendig, das Produkt StP/IM (- *Information Modelling*) getrennt zu lizenzieren. Dieses ist in der Lage, basierend auf den aktuellen Daten des Repositorys eine ER-Modellierung, sowie die Erzeugung von SQL Code für verschiedene Datenbanktypen zur Verfügung zu stellen. StP/IM ist eng mit StP/UML integriert, wodurch eine gemeinsame Nutzung der Informationen sichergestellt wird. Da das Tool jedoch nicht Teil dieser Evaluation ist, wird es nicht näher beschrieben.

g) Diagramm-Ausgabe

Diagramm-Ausgabe, S. 44

1.	*+*
2.	*+*
3.	*-*
4.	*+*
5.	*+*
6.	*+*

StP bietet eine komplett Windows-konforme Druckschnittstelle, die sämtliche, vom Betriebssystem bereitgestellten Einstellungsmöglichkeiten und Dialoge zur Verfügung stellt. Die Druckerausgabe arbeitet nach dem *What You See Is What You Get*-Prinzip. Auf Wunsch kann in Diagrammen direkt der Druckbereich angezeigt werden, eine *Print Preview* Funktion ist nicht implementiert.

Möglichkeiten, den Ausdruck zu konfigurieren, bestehen u.a. im Papierformat, Größe, Hoch-/Querformat und Anpassung an die Seite, kachelartiger Ausdruck sowie weiteren Einstellungen (Filterung des Inhalts etc.).

h) Offenheit - Import / Export

StP bietet verschiedene Schnittstellen zu anderen Produkten:

- ❑ QSS Doors - Anforderungsanalyse und -auswertung
- ❑ Verschiedene Programmierumgebungen (Visual C++, SNiFF, Ada, u.a.)
- ❑ ATA DocEXPRESS - standardisierte (ISO, ANSI/IEEE) Dokumentation
- ❑ Andere Dokumentationstools (siehe *Report-Ausgabe*, S. 132)
- ❑ Produkte der StP Familie (StP/IM, StP/SE, Validator/Req, u.a.)

Als weitere standardisierte Schnittstelle steht *stpem* zur Verfügung, mit deren Hilfe der Zugriff (Import/Export) auf die, im Repository gespeicherten Informationen möglich ist (siehe *Datenhaltung*, S. 130).

Zwischen Diagrammen erfolgt der Austausch von Informationen (z.B. Klassen) gewohnt mittels Cut(Copy)&Paste. Das Einfügen von Texten via Cut&Paste über die Zwischenablage wird von StP ebenfalls anstandslos unterstützt, andere Elemente (wie Grafiken etc.) können jedoch nicht importiert und in ein Diagramm eingefügt werden.

Offenheit - Import / Export, S. 44

1.	+
2.	*0*
3.	*0*
4.	+
5.	+
6.	-
7.	*0*
8.	*0*

i) Code-Generierung

Die im statischen Modell enthaltenen Informationen lassen sich über verschiedene Generatoren in den entsprechenden Zielsprachen (C++, Java, IDL, Ada95 und Forté TOOL) Coderahmen erzeugen.

Die Codeerzeugung wird über spezielle QRL-Scripts geregelt und kann daher mit ein wenig Aufwand sehr flexibel an die entsprechenden Wünsche des Benutzers angepaßt werden. In diesem Zusammenhang sei angemerkt, daß sich die vorliegende Evaluation am C++-Script orientiert, welches dem Programmpaket standardmäßig beiliegt.

Vor Anstoßen der Codegenerierung steht auch ein spezieller Dialog zur Verfügung, in dem allgemeine Angaben getätigt werden können. Dazu gehören unter anderem:

- ❑ Zielverzeichnis für die generierten Dateien
- ❑ Auswahl, ob für jede Klasse eine eigene Datei angelegt werden soll
- ❑ Dateierweiterungen (*.h, *.cpp,...)
- ❑ Auswahl, ob Code für Typen / Interfaces generiert wird
- ❑ Vorlage (*Template*) für Implementierung von Assoziationen

Code-Generierung, S. 45

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	+
8.	+
9.	+
10.	+
11.	+

Die von StP erstellten Coderahmen sind vollständig und überdies nicht mit Kommentaren überfrachtet, was die Lesbarkeit und Verwendbarkeit sehr steigert. Es werden sämtliche Informationen von Attributen, bzw. Operationen übernommen, die in den entsprechenden Class-Tables eingegeben wurden. Dazu gehören neben den üblichen Informationen wie Typ, Parameterliste und Rückgabewert, auch sprachspezifische Angaben (z.B. Sichtbarkeit, Konstante / Variable etc.). Je nachdem welches Template (s.o.) ausgewählt wird, implementiert das Tool Assoziationen zwischen Klassen abhängig von der Kardinalität als einzelnen Zeiger bzw. als Set von Zeigern.

StP setzt die Modellinformationen sehr genau und wohldefiniert um: Assoziationen werden nur dann interpretiert, wenn die Navigierbarkeit zwischen den Klassen definiert ist, Rollennamen dienen als Bezeichner des Zeigers (Sets), sind diese nicht vorhanden, wird ein Defaultbezeichner vergeben (z.B. *ref_B*).

Um Code zu erkennen, der außerhalb des Tools hinzugefügt wurde, bedient sich StP verschiedener Markierungen (Kommentare), um diesen nicht bei einem erneuten Codegenerierungsdurchlauf zu überschreiben. Es kann jedoch auch innerhalb von StP z.B. einer bestimmten Operation Code zugewiesen werden. Dies erfolgt über den sogenannten *Object Annotation Editor*, der den Elementen eines Modells verschiedene spezifische Informationen (in diesem Fall eine Codestück) hinzufügen kann.

Zur Erläuterung folgt ein kleines Beispiel:

Es wurde der C++ Code für eine Klasse A erzeugt, die unter anderem eine Operation *do_something* mit bestimmten Parametern beinhaltet. Im *Object Annotation Editor* wurde dieser eine kurze C++ Anweisung angefügt. StP generiert daraufhin folgende Datei "A.cpp":

```
#include "A.h"
// stp operation 50::51
bool
A::do_something(int x)
{

   // stp code
   return (x>0);
   // end stp code

}
// end stp operation
```

Wie bereits erwähnt, trennt StP die Teile des Codes, die im Tool deklariert wurden (zwischen `//stp code` und `//end stp code`) von denen, die der Benutzer außen (d.h. in einem Editor, bzw. in einer Programmierumgebung) hinzufügt. Dies hilft dem Tool, bereits bestehenden Code bei einer erneuten Durchführung der Codegenerierung nicht zu überschreiben. Jedoch darf der Anwender einerseits die Codemarken nicht entfernen, andererseits wird außerhalb des Werkzeugs hinzugefügter Programmcode überschrieben, falls sich dieser zwischen den besagten Markierungen befindet und nicht vorerst in das Modell importiert wurde.

j) Reverse-Engineering

Reverse-Engineering, S. 45

1.	+
2.	+
3.	-

StP erlaubt das Einlesen von beliebigen Codestücken und das Erstellen eines neuen Modells, bzw. das Updaten eines bestehenden aus den Informationen. Als Programmiersprachen können wie bei der Codegenerierung C++, Java, Ada und IDL verwendet werden. Standardmäßig ist jedoch nur das Reverse Engineering von C(++) Code möglich.

Über das Menü *Reverse Engineering - Parse Source Code...* wird ein Dialog geöffnet, in dem die Dateien angegeben werden können, die StP analysieren soll. Es ist möglich, noch weitere Einstellungen zu definieren, etwa Präprozessor Optionen. Das Tool liest den Code ein und erstellt in einem weiteren Schritt (der eigens vom Benutzer über den Eintrag *Generate Model from Parsed Source Files...* angestoßen werden muß) ein neues Modell auf Basis der erhaltenen Informationen. Hierbei können wieder verschiedene Optionen eingestellt werden (z.B. was soll generiert werden - Diagramme / Tables / Annotationen, u.a.).

StP ist jedoch auch in der Lage, ein Update eines bereits bestehenden Modells durchzuführen (in Hinsicht auf das sog. *Round-Trip Engineering*). Wurde der entsprechende Code eingelesen, so können die Diagramminformationen, bzw. die Klassentabellen über die Funktion *Construct from Reverse Engineering* auf den aktuellen Stand gebracht werden.

Trotz dieser umfangreichen Möglichkeiten fallen auch einige Schwächen des Werkzeugs auf:

- Die Anforderungen an die Ressourcen des Rechners sind überdurchschnittlich hoch, und die dadurch entstehende Performanz mehr als unzureichend. So wurde z.B. im Rahmen dieser Evaluation ein bereits bestehendes C++ Projekt eingelesen. Dieses bestand aus 10 Header-Dateien mit etwa 17 enthaltenen Klassen, was für ein durchschnittliches Projekt relativ wenig ist. Der Reverse Engineering Vorgang benötigte ungefähr 6-7 Minuten, um das betreffende Modell zu erzeugen. Im Vergleich zu anderen Tools ist der Zeitaufwand in Anbetracht des

verwendeten Systems (P2-350, 128MB RAM, Jet Repository) nicht wirklich akzeptabel.

- Durch das konventionelle Reengineering wird kein einzelnes, vollständiges Klassendiagramm erzeugt, sondern mehrere Teildiagramme, die einen bestimmten Ausschnitt des Modells darstellen. Dies soll wiederum anhand eines kleinen Beispiels erläutert werden: Folgende simple C++ Header-Datei wurde analysiert:

```
class Alpha;
class Beta;

class Alpha
{
  public:
  protected:
    Beta* ref_B;
  private:
};

class Beta
{
  public:
  protected:
    Alpha* ref_A;
  private:
};
```

Die Assoziation wird korrekt erkannt und StP erzeugt nun zwei Diagramme, in denen jeweils beide Klassen enthalten sind (siehe Abb. 5–2 und Abb. 5–3), einmal jedoch die Klasse *Alpha* mit einer Assoziationsrolle für *Beta* und umgekehrt.

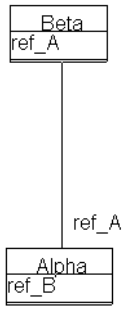

Abb. 5–2
Reverse-Engineering mit StP, Diagramm 1

Abb. 5–3
Reverse-Engineering mit StP, Diagramm 2

Dies wird auch bei anderen Beziehungen von Klassen untereinander ähnlich durchgeführt, so wird etwa für Generalisationen ebenfalls ein eigenes Diagramm erzeugt.

Überdies sind bei genauerer Betrachtung des Beispielmodells weitere Ungereimtheiten zu erkennen. So führt etwa die semantische Prüfung zu dem Ergebnis, daß ein Konflikt zwischen dem Rollennamen und dem in der Klasse enthaltenen Attribut besteht und daher das Modell nicht konsistent ist. Um dies zu überprüfen, wurde ein weiteres Beispielmodell angelegt, das zwei Klassen mit beidseitig navigierbarer Assoziation und der Kardinalität *1 zu 1* enthält. Für dieses wurde zunächst Programmcode erzeugt, der in einem anderen Projekt dann rückwärts generiert wurde. Auch hier erfolgte die gleiche Fehlermeldung der Konsistenzprüfung, obwohl der Code vom Tool selbst erstellt worden war.

k) Test / Verifikation

In StP/UML können keine Testfälle generiert werden. Hierzu ist die Einbindung der Aonix Produkte StP/T, bzw. Validator/Req notwendig, die jedoch nahtlos auf die Modelldaten zugreifen können.

Test / Verifikation, S. 45

1.	*-*
2.	*-*

l) Maßzahlen und Statistiken

Die Erstellung und Ausgabe von Maßzahlen und Statistiken über das aktuelle Modell können in StP nur über Scripts, die auf die Repository Daten zugreifen, abgewickelt werden. Dies bedeutet leider einen zusätzlichen Aufwand für den Anwender, das Ergebnis kann jedoch sehr flexibel gestaltet werden.

Maßzahlen und Statistiken, S. 46

1.	*0*
2.	*0*
3.	*0*
4.	*0*

m) Multi-User - Eigenschaften

StP, das ursprünglich nur für UNIX-Systeme konzipiert war, ist seit der Version 7.0 auch direkt, d.h. ohne X-Emulation auf Windows-NT Computern lauffähig. Jedoch zeigt schon die "Herkunft", daß die Funktionen des Tools auf den Multi-User Betrieb im Netzwerk ausgelegt sind. Folgende Rechner- und Betriebsysteme werden zur Zeit unterstützt:

- ❑ Windows NT (Intel Prozessoren)
- ❑ Sun SPARC, Solaris
- ❑ HP 9000 Serie 700/800, HP-UX
- ❑ IBM RS/6000, AIX
- ❑ Digital Alpha, UNIX

Multi-User - Eigenschaften, S. 46

1.	*+*
2.	*+*
3.	*0*
4.	*+*
5.	*0*
6.	*-*
7.	*-*

Gleich, auf welchem System die Anwender in einer verteilten Entwicklungsumgebung arbeiten, die Modell- und Repository-Daten sind vollständig portierbar.

Die Verwaltung der Benutzer und deren Rechte (*Locking*-Mechanismus) wird von StP vollständig an die zugrundeliegende Datenbank, die sämtliche Modellinformationen hält, delegiert. Die Funktionen von Sybase im speziellen sind hierfür mehr als ausreichend, außerdem wird dadurch auch die konzeptionelle Trennung zwischen Daten und Modellierung als spezielle Sicht auf diese gewährleistet (siehe *Datenhaltung*, S. 130).

Das Tool bietet ausgereifte Funktionen zur Konfliktlösung bei gemeinsamem Zugriff mehrerer Benutzer auf dasselbe Modell. Wie bereits angesprochen, kann ja etwa ein und dieselbe Klasse in mehreren Diagrammen verwendet werden. So wäre es nur schwer möglich, die Daten konsistent zu halten, bzw. vor dem Zugriff anderer Benutzer zu schützen. StP löst dieses

Problem dadurch, daß der Bearbeiter der speziellen Klasse einen sog. *Lock* auf die Tabelle der Klasse setzt (dieser enthält ja sämtliche Informationen über deren Struktur u.a.). Dies wird ohne weitere Angaben automatisch vom Tool durchgeführt, ohne daß der Anwender eigens sich darum kümmern muß. Sobald dieser die Arbeit beendet hat, wird der Lock wieder freigegeben (die Tabelle kann jedoch auch weiterhin gesperrt bleiben). Locks sind in unterschiedlicher Granularität einsetzbar, beginnend auf der Modell, über die Diagrammebene bis auf Klassenebene, welches die feinste Stufe darstellt.

Die Administration (Anzeigen, Löschen, Setzen etc.) von Locks ist in StP über die Funktion *Manage Locks* ebenfalls effizient realisiert. In ähnlicher Weise ist die Verwaltung der Benutzer über den *Repository Manager* möglich. Es können alle aktuellen Benutzer aufgelistet werden, diese können gelöscht oder neue hinzugefügt werden, außerdem können für diese, Paßwörter vergeben und bei Bedarf geändert werden - jedoch nur bei Verwendung der Sybase Datenbank.

n) Security

Security, S. 46

1.	*0*
2.	*+*
3.	*-*
4.	*+*
5.	*+*
6.	*+*
7.	*+*
8.	*0*
9.	*+*
10.	*0*
11.	*+*

Wie bereits in *Multi-User - Eigenschaften*, S. 137 beschrieben, werden sämtliche Aufgaben zur Verwaltung mehrerer Benutzer von der Datenbank (im speziellen nur von Sybase) übernommen.

Es können unabhängig vom Benutzerkonzept des Betriebssystems Benutzer mit unterschiedlichen Rechten ausgestattet werden, diese beschränken sich jedoch grundsätzlich auf die Unterscheidung von Lese- und Schreibzugriff auf Projektdaten. Darüber hinaus können Administratoren definiert werden, die weitergehende Rechte besitzen (Verwaltung der übrigen User, Anlegen / Löschen von Projekten etc.). Die Vergabe der Rechte selbst obliegt dem Datenbankadministrator.

Sicherheitsmaßnahmen, wie Backup, Recovery oder Logging-Mechanismen werden wiederum vollständig vom zugrundeliegenden Datenbanksystem zur Verfügung gestellt und sind auch nur über dieses (d.h. nicht aus StP heraus) durchführbar.

Defaultmäßig wird das Löschen von Elementen nur dann abgefragt, falls es sich um "nicht kritische" Informationen handelt, etwa der Repräsentation einer Klasse in einem Diagramm, im Gegensatz zu deren Tabelle. Der zur Verfügung gestellte Undo-Mechanismus kann eine unbeabsichtigte Löschung jedoch auch wieder rückgängig machen.

o) Multi-Projekt-Eigenschaften / Versionsmanagement

Multi-Projekt-Eigenschaften / Versionsmanagement, S. 47

1.	*+*
2.	*+*
3.	*+*
4.	*0*
5.	*-*

StP ist ohne weiteres in der Lage, mehrere Projekte zu verwalten. Informationen, die in einem Projekt vorkommen, können effizient über eine eigene Import/Export Funktion in anderen Projekten wiederverwendet werden. Hierzu muß das entsprechende Element nur ausgewählt und im Export-Dialog der Name des Ziel-Projekts angegeben werden, das Einrichten von Referenzen auf projektfremde Inhalte ist jedoch nicht möglich, der Austausch erfolgt nur über die erwähnte Funktion, bzw. über das Kopieren von gesamten Diagrammen oder Tabellen.

Eine Versionsverwaltung ist in StP nicht direkt implementiert, jedoch kann diese über die Anbindung an derartige Werkzeuge realisiert werden.

5.7.3 Ergonomie

a) Benutzeroberfläche

Wie bereits beschrieben, war StP in früheren Versionen ein reines UNIX-Produkt und verwendete eine X-Windows Oberfläche, eine Lauffähigkeit auf Windows-NT Systemen war nur über eine Emulation möglich. Dies erforderte natürlich einige Kompromisse in Hinsicht auf die Unterschiede der GUIs mit sich.

Seit der Version 7.0 wurde StP vollständig auf Windows portiert, jedoch können verschiedene Details noch nicht ganz die eigentliche Herkunft verschweigen. So ist eine durchgängige dialogbasierte Schnittstelle, wie sie in modernen Windows-Applikationen üblich ist, nur an wenigen Stellen wirklich effizient implementiert. Der Benutzer wird zwar meist durch einen Dialog geführt, am Ende stehen aber oft nur textuelle Ausgaben zur Verfügung. Dies zeigt sich auch an der überdurchschnittlichen Verwendung von Scripts, Regelfiles u.a., um die Arbeitsweise des Tools zu spezifizieren. Sicherlich ist dies ein Weg, um eine höchstmögliche Offenheit und Anpassungsfähigkeit der einzelnen Funktionen zu erreichen. Jedoch ist es im Projektalltag wohl nur schwer realisierbar, bei geringfügigen Anpassungen auch regelmäßig die verschiedenen Scripts zu ändern, abgesehen vom Aufwand, die entsprechende Syntax der Scriptsprache QRL ausreichend zu erlernen, auch wenn diese sich stark an die C-Syntax anlehnt.

Benutzeroberfläche, S. 47

1.	*0*
2.	*+*
3.	*0*
4.	*0*
5.	*+*
6.	*0*
7.	*0*

Besonders unangenehm fällt in diesem Zusammenhang auch konkret das Fehlen eines, von anderen Werkzeugen gewohnten Repository Browsers auf. Die gewählte Implementation durch Tabellen, die durch Datenbankabfragen mit den Elementen des Repositorys gefüllt werden, ist wohl für ein modernes CASE-Tool nicht ausreichend.

Sonst ist die Benutzeroberfläche jedoch sowohl ansprechend als auch übersichtlich ausgeführt. Die Menüstruktur wirkt wohl durchdacht, es fällt jedoch auf, daß im Kontextmenü nicht immer diejenigen Einträge gefunden werden können, die man erwartet. So kann der Benutzer über das Kontextmenü einer Klasse weder zu deren Klassentabelle oder Zustandsdiagramm navigieren, noch sind verschiedene andere zu erwartende Funktionen aufrufbar, wie etwa die direkte Generierung von Code etc. Diese Einträge finden sich alle in der Menüzeile unter dem Punkt *GoTo*. Dieses Menü besitzt auch je nach ausgewähltem Element einen unterschiedlichen Aufbau - also ein "klassisches" Kontextmenü.

Fortschrittsbalken werden nur bei der Erstellung, bzw. beim Öffnen von Modellen oder Diagrammen geboten. Leider ist diese Funktion nicht auch für andere Bereiche verfügbar (z.B. Reverse-Engineering).

b) Tool-Dokumentation

Tool-Dokumentation, S. 48

1.	*+*
2.	*+*
3.	*-*
4.	*-*
5.	*+*
6.	*+*
7.	*0*
8.	*+*
9.	*+*
10.	*+*
11.	*+*
12.	*+*

Die Dokumentation zu StP, die bis auf einige Ausnahmen als PDF-Files mitgeliefert wird (was bei einem Gesamtausmaß von über 2000 Seiten nicht verwunderlich ist) deckt alle wichtigen Bereiche der Modellierung und der Arbeit mit StP ab. Es ist eine große Anzahl von einzelnen Dokumenten vorhanden, die sehr gut aufgebaut und informativ sind. Leider werden diese nur in englischer Sprache zur Verfügung gestellt. Ein umfangreiches Beispielprojekt liegt dem Programmpaket ebenfalls bei. Die Erstellung desselben wird auch in der Dokumentation Schritt für Schritt beschrieben, ein ungeübter Benutzer kann sich so rasch in die Funktionsweise des Tools einarbeiten.

Besonders hervorzuheben sind auch die UML-spezifischen Dokumente, die nicht nur die Modellierung in StP beschreiben, sondern auch auf inhaltliche und semantische Fragen der UML ausführlich eingehen.

Die Dokumentation sind direkt aus StP heraus aufrufbar (über das Menü *Help*.

c) Usability

Usability, S. 48

1.	0
2.	-
3.	+
4.	+
5.	0
6.	0
7.	+
8.	0
9.	0
10.	0
11.	-

Wie bereits im Zusammenhang mit der Benutzeroberfläche beschrieben, ist die Funktionsweise von StP für unerfahrene (Windows-)Benutzer nicht unbedingt einfach zu verstehen. StP ist definitiv kein Tool, das ohne größere Einarbeitung effektiv bedient werden kann. Jedoch wurde in der aktuellen Windows-Version versucht, die Funktionen des Werkzeugs gut in die Benutzeroberfläche zu integrieren, was auch als durchaus gelungen angesehen werden kann.

Es stehen neben den bereits angesprochenen Kontextmenüs auch andere Standardhilfsmittel, wie Toolleisten für häufig verwendete Befehle oder die Möglichkeit der Verwendung von Tastaturshortcuts zur Verfügung. Diese können auch vom Benutzer (meist wieder über Scripts) an die persönlichen Bedürfnisse angepaßt werden. Leider ist die Fehlerbehandlung etwas unzureichend, der Anwender ist auf die Interpretation einer textuellen Fehlermeldung in einem Log-Fenster beschränkt, weiterführende Informationen zur Ursache des Fehlers werden meist nicht geboten.

StP verwendet so gut wie nie Mehrfenstertechnik innerhalb eines Programmfensters. Statt dessen wird etwa für jedes Diagramm ein eigener Task generiert, wodurch die Windows-Taskleiste bei der Arbeit schnell überfrachtet ist.

d) Hilfefunktionen

Hilfefunktionen, S. 49

1.	-
2.	-
3.	-
4.	+

Das Hilfesystem von StP beschränkt sich vollständig auf die bereits erwähnten PDF-Dokumente. Es gibt keine Hilfefunktion im herkömmlichen Sinn (obwohl dies durch das Menü *Help* ein wenig vorgetäuscht wird).

Dies ist für das Auffinden eines Hilfethemas völlig unzureichend, da die einzige Möglichkeit besteht, die PDF Files nach irgendwelchen Begriffen zu durchsuchen, jedoch ist eine Suche über alle diese Dokumente auch nicht möglich.

e) Graphikeditor / Darstellung

Graphikeditor / Darstellung, S. 49

1.	+
2.	-
3.	+
4.	-
5.	+
6.	0
7.	-
8.	0
9.	+
10.	0
11.	0
12.	-
13.	+
14.	-
15.	0

Die Darstellung der in einem Diagramm enthaltenen Elemente ist in StP sehr gut implementiert. Es kommt praktisch nicht vor, daß sich Elemente (besonders Assoziationen oder andere Beziehungen) überlappen oder überschneiden. Wird eine Klasse etwa über einer anderen abgelegt, so erfolgt eine automatische Anpassung. Linien können entweder als gerade Linie, oder auch treppenartig abgestuft dargestellt werden.

Wird ein Diagramm gespeichert und wieder geladen, bleibt das Layout vollständig erhalten.

Für jedes Element können Defaulteinstellungen hinsichtlich der Größe gemacht werden. Farben können keine eingerichtet werden. Der Ausdruck entspricht eins zu eins der Diagrammdarstellung.

Eine Refresh-Funktion zum Neuzeichnen des Bildschirminhalts ist ebenfalls vorhanden, leider funktioniert diese nicht immer einwandfrei. So ist bisweilen ein mehrfaches Anwählen derselben notwendig, bis "verwaiste" Elemente nicht mehr dargestellt werden.

Das Zoomen des Bildschirminhalts ist in geringer Granularität möglich, außerdem kann etwa der selektierte Teil eines Diagramms oder das gesamte Diagramm auf die sichtbare Zeichenfläche eingepaßt werden.

f) Graphikeditor / Bedienung

Graphikeditor / Bedienung, S. 50

1.	+
2.	+
3.	+
4.	+
5.	+
6.	0
7.	+
8.	+
9.	+
10.	+
11.	0
12.	+
13.	+
14.	+

Die Notationselemente werden aus einer Symbolleiste gewählt und durch einfachen Mausklick auf der Zeichenfläche abgelegt. Außerdem kann durch Doppelklick auf das entsprechende Werkzeug das gleiche Element mehrfach hintereinander angelegt werden, ohne erneut das Symbol auswählen zu müssen. Anschließend kann auf Wunsch direkt im Diagramm ein Bezeichner für das angelegte Element eingegeben werden.

Das Abbrechen einer ungewollten Aktion ist möglich, am einfachsten mit einem Klick der rechten Maustaste oder der Taste *ESC*. Weiters steht eine Undo/Redo Funktion zu Verfügung. Diese kann auch "kleinere" Aktionen rückgängig machen / wiederholen, wie zum Beispiel das Verschieben oder Vergrößern eines Elements. Die Tiefe der Funktion ist defaultmäßig auf fünf Schritte beschränkt, kann aber über eine Anpassung der Regelfiles auf bis zu 99 Schritte erweitert werden.

Der Zeichenbereich ist im Prinzip nicht in der Größe eingeschränkt, StP stellt die gewohnten Scrolling-Mechanismen zur Verfügung. Zur Mehrfachauswahl bestimmter Elemente können diese über eine *Lasso*-Funktion markiert werden. Anschießend kann auf alle Elemente innerhalb der Auswahl eine Aktion durchgeführt werden.

Auch eine Grid-Alignment Funktion zum Ausrichten der Elemente anhand eines Gitternetzes ist vorhanden. Die Größe des Gitters ist ebenso einstellbar. Außerdem ist eine komfortable Alignment Funktion verfügbar, mittels derer mehrere Elemente in Relation zu einander ausgerichtet werden können (oberer, unterer, Rand, horizontal sowie vertikal zentriert, mit konstanten Abständen etc.).

Als weitere gängige Windows-Funktionen wurden Kopieren, Ausschneiden, Einfügen und das Löschen mittels der Taste *Entf* implementiert.

g) Performance

Performance, S. 51

1.	-
2.	0
3.	0
4.	-

Die mangelhafte Performanz ist ein großes Manko von StP. Selbst in Anbetracht der Testrechnerkonfiguration und bei der Verwendung des im Vergleich zu Sybase weitaus anspruchsloseren Repository-Systems Jet sind deutliche Verzögerungen etwa beim Bildschirmaufbau der Diagramme oder beim Verschieben von Elementen zu erkennen. Das Starten der Editoren wird außerdem dadurch verlangsamt, daß eine Reihe von Regelfiles eingelesen und umgesetzt werden müssen.

Auch das Wiedereinlesen von Code und das Erstellen eines Modells aus den enthaltenen Informationen fordert dem Anwender einiges an Geduld ab (siehe *Reverse-Engineering*, S. 135).

h) Adaptivität

Adaptivität, S. 51

1.	-
2.	0
3.	0
4.	+
5.	+
6.	-

Die Oberfläche von StP läßt sich bis auf wenige Ausnahmen (Ein-/Ausblenden von bestimmten Symbolen in den Werkzeugleisten) nicht direkt im Tool anpassen. Grundsätzlich muß immer der Inhalt der entsprechenden Regelfiles verändert werden. Diesem Umstand ist auch ein eigener Teil der Dokumentation gewidmet, was aufgrund der Komplexität auch notwendig ist.

Die Scriptsprache *QRL* (*Query and Reporting Language*) bietet umfangreiche Möglichkeiten zur Definition von Makros u.a. Im Prinzip werden fast

alle Funktionen von StP über QRL definiert und geregelt. Der Sourcecode des Werkzeugs selbst steht dem Anwender jedoch nicht zur Verfügung.

5.8 WithClass 99 Enterprise

WithClass ist zu den eher kleineren Werkzeugen zu zählen, jedoch ist die allgemeine Funktionalität für Projekte bis mittlerer Größe sicherlich ausreichend. Eine wirklich kontrollierte Zusammenarbeit mehrerer Benutzer an einem bestimmten Projekt ist aufgrund der verwendeten Datenhaltung über eine MS-Access Datenbank eher schwierig möglich.

WithClass verhält sich im Zusammenhang mit der Codegenerierung recht flexibel, das statische Modell wird korrekt in Quellcode übersetzt. Auch das Reverse-Engineering funktioniert problemlos. Unbefriedigend ist der hohe Zeitaufwand, den die Codegenerierung benötigt, sowie besonders die etwas chaotische Darstellung der Elemente in den Diagrammeditoren.

5.8.1 Methodenbasierte Kriterien

a) Modellierung der Objektstruktur - Objektstatik

Modellierung der Objektstruktur - Objektstatik, S. 38

1.	*0*
2.	*0*
3.	*-*
4.	*+*
5.	*+*
6.	*+*
7.	*+*

Alle Basiselemente eines Klassendiagramms werden automatisch im Data-Dictionary, welches dem aktuellen Projekt zugrundeliegt, gespeichert. Dieses ist jedoch nicht gleichzusetzen mit dem "Repository", welches die gemeinsame Datenbasis für mehrere Projekte darstellt (wird nachfolgend noch genauer beschrieben). Erst durch eine Export-Funktion werden die Daten des Dictionarys im Repository persistent gespeichert.

Beim Anlegen von neuen Elementen wird jedoch nicht auf Namenskonflikte überprüft. Wird etwa eine Klasse mit bereits vorhandenem Namen nochmals im Diagramm eingezeichnet, so wird im Data-Dictionary ein neues Objekt mit gleichem Bezeichner angelegt und die vorhandenen Attribute und Methoden übernommen. Wird dieses Objekt im Klassendiagramm nachträglich verändert (wie Eintragen von neuen Attributen etc.), so hat dies keinen Einfluß auf das Data-Dictionary.

Die nachträgliche Bearbeitung von korrekt angelegten Objekten im Klassendiagramm funktioniert anstandslos, jedoch können die enthaltenen Elemente nicht umgeordnet werden.

Für Attribute kann neben dem Datentyp ein Initialisierungswert angegeben werden. Weiters ist es möglich, Minimal- und Maximalwerte sowie die Sichtbarkeit anzugeben.

Das Programm läßt es zu, für angelegte Methoden einen Rückgabetyp, Vor- und Nachbedingungen, sowie eine Parameterliste mit Datentypen der einzelnen Parameter einzutragen. Außerdem wird eine Auswahl von bestimmten Schlüsselwörtern für Operationen (z.B. *virtual*, *abstract*, *static*,...) zur Verfügung gestellt.

b) Modellierung der Systemstruktur - Systemstatik

Beziehungen können nachträglich bearbeitet werden, Partner können einfach gewechselt werden (per Drag&Drop) und auch die Art der Beziehung kann in einem Dialogfenster z.B. von *Assoziation* auf *Aggregation* geändert werden. Die Einteilung von Klassen in andere Kategorien als Packages wird nicht unterstützt, ebenso können OCL-Annotationen bei Zusicherungen und Einschränkungen nur in einem Textfeld eingetragen werden.

Modellierung der Systemstruktur - Systemstatik, S. 39

1.	+
2.	+
3.	-
4.	0

c) Dynamisches Objektverhalten - Objektdynamik

Zustandsdiagramme können angelegt werden, haben jedoch nur eingeschränkt Bezug zu anderen Sichten des Modells. Dies äußert sich vor allem darin, daß nicht von vornherein ein Zustandsdiagramm für eine bestimmte Klasse angelegt werden kann, sondern daß in einem neuen Diagramm sämtliche Zustände einzeln der Klasse zugeordnet werden müssen.

Zustandsübergänge können mit Bedingungen verknüpft werden, da diese jedoch nur in einem einfachen Textfeld angegeben werden, kann kein Bezug auf Attribute genommen werden. Genauso verhält es sich mit Aktionen, die ebenso textuell beschrieben und einem Zustandsübergang zugeordnet werden, Bezug auf Methoden eines Objekts jedoch nicht zulassen. Geschachtelte Zustandsdiagramme können erstellt werden. Die Modellierung einer Generalisation von Ereignissen ist nicht möglich.

Dynamisches Objektverhalten - Objektdynamik, S. 39

1.	0
2.	0
3.	+
4.	0
5.	0
6.	-

d) Dynamisches Systemverhalten - Systemdynamik

Das Erstellen von Sequenzdiagrammen wird von WithClass ausreichend unterstützt. Innerhalb von Nachrichten können vorhandene Methoden des aufgerufenen Objekts ausgewählt werden, auch die Angabe von aktuellen Parametern ist möglich. Ist der Bezeichner einer Nachricht nicht in der Klasse des entsprechenden Objekts als Methode enthalten, so wird abgefragt, ob ein neuer Eintrag automatisch vorgenommen werden soll. Leider dürfte in diesem Zusammenhang ein Programmierfehler im Werkzeug vorhanden sein: Wird das automatische Eintragen der neuen Methode vom Benutzer abgelehnt, so wird im Laufe der folgenden Bearbeitung des Sequenzdiagramms dies jedesmal wiederholt abgefragt.

Dynamisches Systemverhalten - Systemdynamik, S. 39

1.	0
2.	0

e) Zusammenhang zwischen den Modellen

Ein Zusammenhang zwischen den Modellen ist nur teilweise gegeben, da nicht ein Zustandsdiagramm selbst, sondern nur die darin enthaltenen Zustände einzeln bestimmten Klassen zugeordnet werden können (siehe *Dynamisches Objektverhalten - Objektdynamik*, S. 143). So wird auch das dynamische Verhalten nicht an etwaige Unterklassen vererbt. Ereignisse können zwar näher spezifiziert werden, aber sie stehen in keiner Beziehung zu Methoden der entsprechenden Klasse.

Zusammenhang zwischen den Modellen, S. 39

1.	0
2.	-
3.	0
4.	-

f) Abstraktionsebenen / Sichten

Es besteht die Möglichkeit, über verschiedene Buttons den Detaillierungsgrad der Darstellung im Klassendiagramm festzulegen. Der Anwender kann etwa auswählen, ob nur Klassennamen, oder zusätzlich z.B. die Attribute, eventuell zusammen mit Typen, Parametern etc. angezeigt werden sollen. Jedoch kann die Sichtbarkeit von Komponenten nicht zur Einstellung heran-

Abstraktionsebenen / Sichten, S. 40

1.	+
2.	+
3.	0
4.	-
5.	0

gezogen werden. Eine Default-Einstellung kann nicht getroffen werden, die jeweils aktuelle Einstellung einer Datei wird jedoch mitgespeichert.

g) Vorgangsreihenfolge bei der Entwicklung

Vorgangsreihenfolge bei der Entwicklung, S. 40

1.	+
2.	-
3.	0
4.	-

Eine bestimmte Reihenfolge bei der Erstellung eines Projekts wird von WithClass nicht verlangt. Ein Anlegen von Elementen ist nur in einem Diagramm direkt möglich, d.h. diese können nicht erst im Data-Dictionary angelegt und später im Diagramm übernommen werden. Eingezeichnete Elemente werden automatisch ins Data-Dictionary übernommen und dort gespeichert.

h) Unterstützung bei der Methodenanwendung

Unterstützung bei der Methodenanwendung, S. 40

1.	0
2.	0
3.	0
4.	0
5.	-
6.	0

Die Unterstützung bei der Anwendung der UML im Rahmen der Hilfe beschränkt sich auf eine kurze Einführung in die Erstellung von UML-Diagrammen mit Hilfe von WithClass. Die Elemente der UML werden nicht näher erläutert.

Allgemein entsprechen die Notationselemente annähernd jenen der UML, sodaß meist keine Verwirrung beim Betrachten herrscht.

Bei der Verletzung von Konsistenz- und Integritätsbedingungen wird weder automatisch gewarnt noch werden diese weiterführend erläutert (z.B. nach einem Fehlerreport).

5.8.2 Funktionale Kriterien

a) Datenhaltung

Datenhaltung, S. 41

1.	0
2.	0
3.	-
4.	-
5.	0
6.	-
7.	0
8.	0

Ein Projekt in WithClass besteht aus zwei Files für jedes Diagramm. Die Informationen aus jedem Teilmodell werden über ein gemeinsames Repository synchronisiert, das in Form einer MS-Access Datenbank ("STANDARD.MDB") abgelegt wird.

Modellinformationen werden im Repository nichtredundant gespeichert. Wird zum Beispiel versucht, zwei Klassen mit demselben Namen im Klassendiagramm anzulegen, ist dies zunächst möglich, beim Export in das externe Repository liefert das Programm jedoch eine Warnung und übernimmt die Klasse nur einmal.

Werden Daten aus dem Repository in ein neues Diagramm importiert, wird automatisch auch das dazugehörige Klassendiagramm gezeichnet, löscht man eine Klasse aus dem Klassendiagramm, wird sie auch aus der Datenbasis entfernt. Modellinformationen sind also nur dann in der Datenbasis enthalten, wenn sie auch in einem (Klassen-) Diagramm existent sind. WithClass bietet keine Möglichkeit, Daten im Repository zu bearbeiten oder zu löschen.

Das Werkzeug speichert Diagramme im *.omt Format, das nicht von anderen Programmen interpretiert werden kann. Das externe Repository wird im Access-Datenbankformat gespeichert, und kann somit auch von anderen Programmen gelesen werden.

WithClass legt beim Speichern eines Diagramms eine Datei im ASCII-Format an, in dem alle relevanten Informationen abgelegt werden. Diese Datei wird vom Programm gelesen, und dieses stellt das Diagramm wieder her.

b) Browsing-Eigenschaften

Browsing-Eigenschaften, S. 41

1.	0
2.	-
3.	0
4.	0
5.	-
6.	0
7.	+
8.	-
9.	-
10.	-
11.	-

WithClass bietet die Möglichkeit, den Client-Bereich des Ansichtsfensters zu splitten, wobei dann im rechten Teil der Klassen-Browser angezeigt wird. Hier wird die Datenbasis in drei Ebenen hierarchisch als Baumstruktur dargestellt, angefangen bei Packages über Klassen zu deren Attributen und Methoden. Es kann zwischen drei Ansichten gewechselt werden:

- ❑ Nur Elemente des aktuell geöffneten Diagramms
- ❑ Elemente aller Diagramme des aktuellen Projekts
- ❑ Elemente, die im gemeinsamen Repository gespeichert sind

Dies ermöglicht ein relativ bequemes Browsen und Auffinden von gesuchten Elementen.

Besitzt dieser Browser den Eingabefokus, kann durch die Eingabe von Zeichen über die Tastatur zu dem nächsten Objekt mit übereinstimmendem Namen gesprungen werden. Wird eine bestimmte Klasse im Browserfenster angewählt, wird die Diagrammansicht auf diese zentriert. Etwas verwunderlich ist jedoch, daß Elemente nur mittels ihres Anfangsbuchstabens aufgefunden werden können, wenn sie sichtbar sind. D.h., daß z.B. bestimmte Attribute einer Klasse nur dann gefunden werden, wenn in der Baumstruktur der entsprechende Zweig geöffnet ist. Dies erscheint ein wenig umständlich. Weitere Funktionen, wie die Suche nach mehreren Objekten mittels spezifizierbarer Filter werden von WithClass nicht zur Verfügung gestellt.

Durch Auswahl von Objekten im Browser gelangt der Benutzer zu denselben Dialogen, die auch zur Bearbeitung in der Diagrammansicht zur Verfügung stehen. Weitere Editierfunktionen wie Löschen, Kopieren, Ausschneiden oder Einfügen werden jedoch nicht angeboten.

Klassenhierarchien werden nicht als solche im Browser dargestellt. WithClass liefert keine Auskünfte darüber, in welchen übergeordneten Strukturen ein bestimmtes Modellierungselement vorkommt.

c) Navigation im Systemmodell

Navigation im Systemmodell, S. 42

1.	+
2.	0
3.	-
4.	0

Die Navigation zwischen verschiedenen Darstellungen des Modells gestaltet sich in WithClass relativ einfach, da in einem eigenen Fenster ein Browser, die sogenannte *Project Information* zur Verfügung steht, mit Hilfe derer man die einzelnen Diagramme bequem anwählen und bei Bedarf neue anlegen kann. Verbindungen zwischen zusammengehörigen Diagrammen, welche die Navigation erleichtern könnten, werden jedoch nicht bereitgestellt.

d) Konsistenzprüfung / Integrität

Konsistenzprüfung / Integrität, S. 42

1.	-
2.	-
3.	0
4.	-
5.	0
6.	-
7.	0
8.	-
9.	0

WithClass läßt es wohl zu, daß ungültige Konstrukte erstellt werden. So ist es z.B. möglich, auch leere Klassen anzulegen (d.h. Klassen ohne Bezeichnung), diese werden als vollwertige Klassen interpretiert. Sie werden ins Data-Dictionary eingetragen, auch Relationen zwischen ihnen und anderen Elementen sind möglich. Relationen (Assoziationen, Datenflüsse etc.) müssen jedoch zwischen zwei Elementen bestehen, sie können nicht ins Leere gehen. Auch können rekursive Vererbungsstrukturen angelegt werden. Mehrere Elemente mit gleichem Namen können eingezeichnet werden (siehe *Modellierung der Objektstruktur - Objektstatik*, S. 142). Erst nach einer erfolgten Konsistenzprüfung werden diese Fehler erkannt und angezeigt.

Wird ein Element aus dem Diagramm gelöscht, so wird es nicht automatisch im zugrundeliegenden Data-Dictionary ebenfalls gelöscht. Dies ist aber nur deshalb der Fall, da das Data-Dictionary nicht automatisch gespeichert

wird. Erst über einen Import-Export-Dialog werden die einzelnen Elemente eines Diagramms (sprich Klassen) im sog. *Class-Repository*, und damit in der Access-Datenbank gespeichert. So können auch mehrere Dateien auf ein gemeinsames Repository zurückgreifen und somit verschiedene Sichten auf das Modell darstellen (z.B. Sequenzdiagramm, Use-Case Diagramm etc.), mit der Einschränkung, daß nach Neuanlegen einer Datei und anschließendem Import der Daten immer das Klassendiagramm automatisch erstellt wird, sodaß etwa das Sequenzdiagramm nicht "allein" in einer Datei dargestellt werden kann.

e) Report-Ausgabe

Report-Ausgabe, S. 43

1.	+
2.	+
3.	+
4.	*0*
5.	*0*
6.	+
7.	-
8.	-
9.	+
10.	*0*

Für die Generierung von Reports stehen unter WithClass verschiedene Scripts bzw. VBA-Makros zur Verfügung, die auch nachträglich an entsprechende Bedürfnisse und Anforderungen angepaßt werden können. Reports werden unter den drei Kategorien *Class-*, *State-* und *Object Code/Report* zusammengefaßt, wobei der Class-Report eine Aufstellung der statischen Elemente (Klassen, Attribute, Methode etc.) generiert, während sich die beiden anderen Reports das dynamische Modell beschreiben (State-Diagramm bzw. Sequenz-Diagramm). Die Reports werden defaultmäßig als ASCII-Textfiles erzeugt, nur die Aufstellung der Klassen etc. läßt sich direkt als HTML-File erzeugen. Hierbei werden eine Übersichts- (nur Klassennamen) und eine Detail-Datei (Klassen mit zugehörigen Elementen, wie Attribute, u.ä.) angelegt.

Weitere Ausgabeformate werden nicht unterstützt. Ein Text-Fehlerreport, der eine Konsistenzprüfung enthält, wird bei einem Check-Vorgang automatisch miterzeugt.

f) Datenbankschema-Ausgabe

Datenbankschema-Ausgabe, S. 43

1.	+
2.	-
3.	-
4.	+
5.	-
6.	+

Aus den wichtigsten Informationen des statischen Modells kann WithClass über definierte Scripts ein relationales Datenbankschema generieren. Hierbei wird für jede Klasse ein Textfile mit SQL-Statements (CREATE TABLE etc.) erzeugt, entsprechende Spezifikationen der Attribute (z.B. als *Primary-Key*, *Not Null* etc.) werden ebenfalls im SQL-Code übernommen, nicht jedoch Beziehungen zwischen Klassen. Ein objektorientiertes Datenbankschema kann jedoch nicht generiert werden.

Die Erstellung von eigenen EER-Diagrammen ist in WithClass nicht möglich.

g) Diagramm-Ausgabe

Diagramm-Ausgabe, S. 44

1.	+
2.	-
3.	-
4.	+
5.	-
6.	*0*

Die Ausgabe von Diagrammen auf Druckern gestaltet sich relative einfach, da der gesamte Prozeß den Windows-internen Funktionen übertragen wird. So ist der Ausdruck auch nicht weiter konfigurierbar, außer, daß wahlweise das gesamte Diagramm oder nur eine bestimmte Region ausgegeben werden können.

Qualität und Schnelligkeit des Vorgangs sind nur über die Drucker-Einstellungen konfigurierbar, das Tool selbst verfügt über keine weiteren Features. Das ausgegebene Dokument selbst entspricht genau der Darstellung auf dem Bildschirm.

h) Offenheit - Import / Export

Offenheit - Import / Export, S. 44

1.	0
2.	0
3.	-
4.	+
5.	0
6.	-
7.	-
8.	0

WithClass kann als relativ offenes Tool angesehen werden, obwohl der Hersteller Microgold selbst kaum Zusatzprodukte anbietet. Jedoch ist es möglich durch Scripts bzw. sogenannte *Add-In Features* das Tool seinen eigenen Bedürfnissen entsprechend anzupassen und es zu erweitern.

In der neuen Version 99 (6.0) können auch mit Rational Rose erstellte Diagramme direkt importiert und interpretiert werden. Weiters ist es möglich, bestimmte Bereiche der Diagramme wahlweise als GIF, JPEG, BMP oder TIF-Grafikdatei abzuspeichern oder per OLE zu exportieren. Ein Import von fremden Grafiken ist jedoch nicht möglich. *Cut and Paste* innerhalb des Tools und auch zwischen dem Tool und anderen Anwendungen ist nur von Text möglich, wobei dies nur in Dialogfenstern selbst unterstützt wir und nicht direkt bei Text-Eingaben im Diagramm (z.B. Anlegen von Klassennamen oder Anmerkungen).

Besonders muß die in der neuen Version erstmals enthaltene Möglichkeit des *VisualBasic for Applications*-Scriptings hervorgehoben werden. Hierdurch können vom Benutzer verschiedene Wizards, User-GUIs und ActiveX-Controls direkt erstellt und angepaßt werden, die so eine sehr flexible und leicht erweiterbare Plattform für benutzerspezifische Anforderungen an während der Arbeit wiederkehrende Aufgaben in WithClass darstellen.

Eine weitergehende Anbindung an andere Programme, wie Textverarbeitungen etc. wird durch die bereits enthaltenen VBA-Makros sichergestellt. So können Klassenreports als MS-Word oder Excel-Datei ausgegeben werden.

i) Code-Generierung

Code-Generierung, S. 45

1.	+
2.	+
3.	+
4.	+
5.	+
6.	+
7.	+
8.	+
9.	0
10.	+
11.	+

Bei der Generierung von Code aus dem statischen Objektmodell verhält sich WithClass relativ flexibel. Die Codegenerierung für verschiedene gewünschte Zielsprachen wird durch benutzerdefinierbare Scripts angestoßen. Es werden vom Tool bereits Scripts für die Zielsprachen bzw. Entwicklungsumgebungen Ada, C++ (ANSI, Borland, Visual), Delphi, Eiffel, IDL, Java, OO Cobol, SQL, Smalltalk und Visual Foxpro bereitgestellt, die noch nachträglich an individuelle Anforderungen angepaßt werden können. Die Scriptsprache selbst gestaltet sich eher einfach, sodaß sich eine entsprechende Definition nicht besonders schwierig ist.

Die Codegenerierung, die für die Sprache C++ mittels der bereits enthaltenen Scripts getestet wurde, macht einen sehr vollständigen Eindruck. Sämtliche Elemente des Klassendiagramms werden in den Code übersetzt, sowohl Attribute inklusive ihrer Typen und Defaultwerten als auch Methoden mit ihren Argumenten und Rückgabetypen sowie Zugriffsbeschränkungen und Beziehungen zwischen Klassen, die korrekt mittels Zeigern umgesetzt werden.

Eine entsprechende Arbeitserleichterung stellt die Möglichkeit dar, innerhalb der Methodendefinition eigene Programmstücke zu codieren, die dann bei der Code-Generierung umgesetzt werden. Hierfür steht ein eigener einfacher Editor zur Verfügung, der auch Schlüsselwörter farbig hervorheben kann und so sehr zur guten Lesbarkeit des Codes beiträgt. Durch einen *Export C++*-Button wird bereits erzeugter Code auf den aktuellen Stand gebracht, sodaß nicht der gesamte Generierungs-Prozeß eigens initiiert werden muß. Wird jedoch der Code außerhalb des Tools verändert, so wird dieser hierbei ohne Warnung überschrieben. Um dies zu vermeiden, muß der Code vorher mittels *Import C++* in den Editor übernommen werden. Auch dies wird ohne Komplikationen durchgeführt.

j) Reverse-Engineering

Reverse-Engineering, S. 45

1.	+
2.	+
3.	+

Das Reverse-Engineering gestaltet sich ähnlich einfach wie auch vollständig wie die Code-Generierung. Das Programm liest das gewünschte Header-File (im Falle C++) und erzeugt anstandslos die entsprechende Klasse im Klassendiagramm, inklusive Attributen und Methoden. Auch hier werden Typen, Parameter u.ä. übernommen. Auch vorhandener Code innerhalb von Methoden kann eingelesen und innerhalb des vorhandenen Editors nachbearbeitet bzw. wieder generiert werden.

k) Test / Verifikation

Test / Verifikation, S. 45

1.	-
2.	-

Eine Möglichkeit zum Testen bzw. zur Simulation auf Modellebene wird von WithClass nicht unterstützt.

l) Maßzahlen und Statistiken

Maßzahlen und Statistiken, S. 46

1.	-
2.	-
3.	-
4.	-

Das Berechnen von Maßzahlen und Statistiken auf Basis des Repositorys wird nicht direkt zur Verfügung gestellt. Es wäre unter Umständen möglich, in MS-Access oder Excel eine selbst erstellte Berechnung anzustoßen, da das Repository ja als Access-Datenbankfile zur Verfügung steht, bzw. verschiedene Exportfilter benutzt werden können.

m) Multi-User - Eigenschaften

Multi-User - Eigenschaften, S. 46

1.	0
2.	-
3.	-
4.	0
5.	0
6.	-
7.	-

WithClass ist zur Zeit nur für die Betriebssysteme Windows 9x bzw. NT verfügbar. Dadurch ist der Netzwerkbetrieb auch nicht auf Multi-User-Plattformen wie UNIX o.ä. möglich.

Die konkrete Arbeit im Mehrbenutzerbetrieb sieht dementsprechend folgendermaßen aus: In einem Windows-Netzwerk wird das Repository (d.h. die Access-Datenbank "STANDARD.MDB") auf einem NT-Server in einem Ordner gespeichert, auf den von mehreren Benutzern zugegriffen werden kann. Jeder User arbeitet somit mit der gleichen Datenbasis und importiert bzw. exportiert sein Modell über dieses Repository. Es ist jedoch auch möglich, auf Package- und Klassenebene den Zugriff auf bestimmte Objekte zu sperren, d.h. diese mit einem *Lock* zu belegen. Hierzu wird von WithClass der Benutzername als Umgebungsvariable mitgespeichert, und so ist nur dieser bestimmte Benutzer berechtigt, ein Update auf die von ihm gesperrten Elemente durchzuführen. Es können jedoch, wie bereits erwähnt, immer nur auf Klassenebene Locks durchgeführt werden. Weitere Multi-User Features (wie Dialogfunktionen u.ä.) werden nicht geboten.

n) Security

Security, S. 46

1.	-
2.	-
3.	-
4.	-
5.	-
6.	-
7.	-
8.	0
9.	0
10.	0
11.	+

Wie schon in *Multi-User - Eigenschaften*, S. 148 beschrieben, wurde die Abwicklung des Mehrbenutzerbetriebs in WithClass nicht besonders umfangreich implementiert. So ist über das *Locking* auf Klassenebene hinaus keine weitere Verwaltung von Benutzern und deren Zugriffsrechten direkt aus dem Tool heraus möglich. Weder müssen sich Anwender vor der Arbeit am Tool anmelden, noch ist ein Zurückverfolgen von Änderungen mittels eines Log-Protokolls möglich.

Vor dem Löschen von Modellelementen wird zuerst mit Hilfe eines Dialogs nachgefragt, ob dieser Vorgang durchgeführt werden soll und auch nach dem Löschen steht eine Undo-Funktion zur Verfügung, um Elemente wiederherzustellen. Es gibt jedoch keinen weitergehenden Recovery-Mecha-

nismus, sodaß ein oftmaliges Speichern des Modells den einzigen Schutz vor Systemabstürzen oder anderen unvorhergesehenen Unterbrechungen bietet.

o) Multi-Projekt-Eigenschaften / Versionsmanagement

Es können in mehreren Fenstern mehrere Projekte gleichzeitig offengehalten und auch bearbeitet werden. WithClass entscheidet je nach aktivem Fenster, welches Projekt gerade bearbeitet wird. Die Modellinformationen werden alle über das gemeinsame Repository "STANDARD.MDB" verwaltet. Sollen bereits bestehende Informationen in einem anderen Projekt wiederverwendet werden, so müssen diese aus dem Repository importiert werden. Ein Versionsmanagement von Modellen ist durch die direkte Anbindung an den Merant PVCS Version Manager einfach und effizient möglich.

Multi-Projekt-Eigenschaften / Versionsmanagement, S. 47

1.	*0*
2.	*0*
3.	*-*
4.	*-*
5.	*-*

5.8.3 Ergonomie

a) Benutzeroberfläche

Die Benutzeroberfläche kann als durchaus gelungen beurteilt werden. Sie entspricht im Großen und Ganzen denen anderer Windows-Programme, enthält die üblichen Elemente (Menüs, Buttons), die sich in ihrem Erscheinungsbild an den üblichen Standards orientieren. Die Übersichtlichkeit ist trotz der relativ großen Anzahl von Auswahlmöglichkeiten (vor allem bei Buttons) bei ausreichend großer Bildschirmauflösung jedenfalls gegeben. Auch Dialoge sind nicht überfrachtet und bieten dennoch alle wichtigen Eingabefelder in einem einzigen Dialog, sodaß nicht zu oft zwischen verschiedenen Reitern gewechselt werden muß. Die Menübefehle sind ebenfalls logisch gegliedert, momentan nicht verfügbare Menübefehle sind, dem Windows-Standard entsprechend, in heller Farbe dargestellt.

Eine Statusanzeige bei länger dauernden Aktionen (etwa Fortschrittsbalken) wird leider nicht geboten, jedoch wird z.B. bei der Codegenerierung in der Statuszeile des Programms angezeigt, welche Klasse sich zur Zeit in Bearbeitung befindet, was einen gewissen Aufschluß über die noch zu erwartende Dauer gibt.

Benutzeroberfläche, S. 47

1.	*0*
2.	*0*
3.	*0*
4.	*+*
5.	*+*
6.	*-*
7.	*0*

b) Tool-Dokumentation

Die eigentliche Dokumentation zu WithClass steht in zwei Teilen zur Verfügung: Einerseits die recht umfangreiche Online-Hilfe, auf der anderen Seite das als eher spartanisch anzusehende Handbuch, das als Acrobat PDF-Datei auf der Webseite des Herstellers zum Download bereitsteht.

Die aus dem Tool aufrufbare Hilfe bietet alle wesentlichen Informationen zum Installieren, Konfigurieren und Arbeiten mit WithClass, sämtliche Funktionen und Dialoge werden für die tägliche Arbeit ausführlich genug beschrieben. Das Handbuch selbst, das ja eigentlich als tiefergehende Referenz dienen sollte, geht nicht ausreichend auf Details ein, sondern bietet nur einen sehr groben Überblick über die Funktionen. So war es etwa bei dieser Evaluation vonnöten, mittels Online-Hilfe und Handbuch abwechselnd zu versuchen, die gewünschte Information zu erlangen. Ein Methodenhandbuch wird ebenfalls nicht geboten, die hinter dem Modell stehende Semantik der Modellierung wird nicht beschrieben. Nur auf ihrer Webseite bietet der Hersteller Microgold sog. Tutorials für die Erstellung von C++, Java und Delphi-

Tool-Dokumentation, S. 48

1.	*0*
2.	*0*
3.	*-*
4.	*-*
5.	*-*
6.	*0*
7.	*0*
8.	*0*
9.	*+*
10.	*+*
11.	*0*
12.	*+*

Projekten und eine Fallstudie zur Verfügung, die jedoch teilweise aus dem Jahre 1995 stammen (C++ Tutorial). Weiters wird hier auch eine UML-Dokumentation bereitgestellt, die sich aber als *Documentation Set* von Booch et al. herausstellt, jedoch leider nur die Version 0.91 der UML aus dem Jahre 1996 beschreibt und somit nicht mehr als aktuell gesehen werden kann. Das einzig wirklich Nützliche stellt ein *Object Oriented Programming Glossary* dar, der die wichtigsten Begriffe erklärt.

Dem Tool beigefügt ist ein fertiges Beispiel, anhand dessen die grundlegenden Modellierungsfunktionen nachvollzogen werden können.

c) Usability

Usability, S. 48

1.	0
2.	-
3.	+
4.	0
5.	+
6.	0
7.	+
8.	0
9.	+
10.	+
11.	+

WithClass ermöglicht im Allgemeinen ein rasches und effizientes Arbeiten. Die übersichtliche Gliederung und vor allem die ausreichende Anzahl an Buttons und Tastatur-Shortcuts läßt den Anwender schnell auf die von ihm gewünschten Funktionen zugreifen. Besonders gut gelungen sind in diesem Zusammenhang die Dialoge bei der Klassenerstellung und -bearbeitung: Es erscheint ein Dialog mit verschiedenen Reitern für klassenspezifische Informationen, Attribute, Methoden, Spezifikationen etc. Gerade Attribute und Methoden können hierbei sehr einfach vorab nur mit ihrem Namen eingetragen werden, die genauere Beschreibung (wie Aufrufparameter, Typen, u.ä.) kann dann später in eigenen Dialogen folgen. So ist es möglich, sehr schnell ein Grundgerüst des statischen Modells anzulegen, ohne sich bereits um nähere Spezifikationen kümmern zu müssen.

Die Fehlerbehandlung entspricht den allgemeinen Standards, da aber eine Konsistenzprüfung des Modells nicht automatisch durchgeführt wird, ist es fast nicht möglich, fehlerhafte Aktionen auszuführen. Wird diese jedoch angestoßen, so werden keine weiterführenden Informationen zu etwaigen Fehlern gegeben.

Die Symbolleisten selbst sind nicht konfigurierbar, es können weder Buttons entfernt, noch welche hinzugefügt werden. Jedoch besteht die interessante Möglichkeit, ein Menü selbst mit neuen Menüpunkten zu belegen, und zwar das Menü *Tools*. Hier können von WithClass aus aufrufbare Programme angegeben werden (in Form von Bezeichnung, EXE-Datei und allfälligen Startparametern), welches aber aufgrund des Windows-Startmenüs nicht unbedingt ein notwendiges Feature darstellt.

Übersetzungen der englischen Originalversion stehen zur Zeit für Deutsch, Russisch, Portugiesisch, Italienisch, Französisch, Spanisch und Japanisch zur Verfügung.

d) Hilfefunktionen

Hilfefunktionen, S. 49

1.	+
2.	0
3.	0
4.	+

Über den Shortcut F1 ist standardmäßig die Online-Hilfe aufrufbar, die als Windows-Helpfile organisiert die üblichen Funktionen, wie Index und Suche zur Verfügung stellt. Weiters ist auch eine *Hot-Help* verfügbar, mit Hilfe derer bequem Informationen über bestimmte Bildschirmelemente abrufbar sind.

e) Graphikeditor / Darstellung

Die Darstellung der graphischen Elemente eines Diagramms selbst ist der große Schwachpunkt von WithClass. Es ist fast unmöglich, das Layout dahingehend zu beeinflussen, daß ein ansehnliches Diagramm entsteht. Besonders Beziehungen zwischen Klassen (Assoziationen, Generalisationen etc.) stellen das Tool anscheinend vor unlösbare Aufgaben in der Darstel-

lung. Diese überschneiden einander fast grundsätzlich, führen sichtbar durch Klassen hindurch u.ä., ohne daß dies in effizienter Weise vom Benutzer beeinflußt werden könnte (siehe hierzu auch Abbildungen in Kapitel 6).

Eine automatische Layout-Funktion fehlt ebenso. So ist es nur mit viel Fingerspitzengefühl und Mühe möglich, eine einigermaßen ansprechende Darstellung zu erhalten. Dies mindert den sonst eher guten Gesamteindruck von WithClass beträchtlich, besonders, da das Tool bereits in einer fortgeschrittenen Version (6.0) zur Verfügung steht.

Abgesehen von dieser Schwäche bietet der Grafikeditor jedoch alle erforderlichen Funktionen. Es können fast alle Symbole der UML dargestellt werden, neben diesen werden auch weitere Formen wie Rechteck, Ellipse etc. zur Verfügung gestellt. Bestimmten Elementen (Klassen, Beziehungen, Zuständen, u.a.) können defaultmäßig bestimmte Farben zugeordnet werden, die später auch für einzelne geändert werden können. Auch die Stärke von Linien und die Schrift (Art, Große,...) können eingestellt werden. Eine Zoomfunktion ermöglicht, den Bildschirminhalt im gewünschten Maße zu vergrößern, bzw. zu verkleinern. Das Layout eines gespeicherten Diagramms wird beim Öffnen beibehalten. Die Größe von Notationselementen ist jedoch nicht einstellbar, sie entspricht immer der, für die Darstellung eines Elementes erforderlichen Mindestgröße (z.B. bei Klassen). Eine Refresh-Funktion des Bildschirminhalts gibt es ebenfalls nicht.

Graphikeditor / Darstellung, S. 49

1.	0
2.	+
3.	+
4.	+
5.	0
6.	0
7.	-
8.	0
9.	+
10.	0
11.	0
12.	-
13.	0
14.	+
15.	-

f) Graphikeditor / Bedienung

Die Bedienung des Grafikeditors erfolgt im Gegensatz zur Darstellung intuitiv und relativ einfach. Da alle Notationselemente mit Hilfe von Buttons auswählbar sind (zusätzlich zur Auswahl in einem Menü), kann schnell die gewünschte Struktur eingezeichnet werden. Hierbei wird das Element an seinem gewünschten Platz eingefügt, nachfolgend kann es noch beschriftet werden. Dies erfolgt ein einem Arbeitsschritt. Leider muß, etwa beim Anlegen von mehreren Klassen hintereinander, immer wieder der entsprechende Button erneut ausgewählt werden.

Elemente eines Diagramms können einfach per Drag and Drop verschoben werden, sind mehrere ausgewählt (durch Umrahmung mit dem Mauscursor), so können diese gemeinsam verschoben werden. Leider funktioniert Undo bei derartigen Aktionen nicht. Der Zeichenbereich für ein Diagramm ist unbegrenzt. Außerdem verfügt das Tool über eine Funktion zum Ausrichten von Elementen anhand eines Rasters (*Grid*). Cut&Paste wird ebenfalls unterstützt, Elemente können auch einfach entfernt werden.

Graphikeditor / Bedienung, S. 50

1.	+
2.	+
3.	-
4.	+
5.	+
6.	+
7.	-
8.	0
9.	0
10.	+
11.	+
12.	0
13.	0
14.	+

g) Performance

Auf dem Testrechner war ein durchgehend zügiges Arbeiten möglich. WithClass dürfte jedoch an die Hardware des Computers keine zu hohen Anforderungen stellen, da laut Dokumentation als Mindestkonfiguration ein 386er Prozessor (empfohlen: 486er) angegeben wird. Die graphische Manipulation von Modellelementen und auch Funktionen wie Zoomen und Scrollen geschehen flüssig und ohne auffällige Verzögerung. Auch das Reverse-Engineering und das Laden und Speichern von Projekten werden in annehmbarem Tempo durchgeführt, nur die Codegenerierung beansprucht übermäßig viel Zeit.

Performance, S. 51

1.	+
2.	+
3.	+
4.	-

h) Adaptivität

Adaptivität, S. 51

1.	-
2.	-
3.	-
4.	*0*
5.	*0*
6.	-

Sehr große Anpassungsmöglichkeiten darf man von WithClass nicht erwarten. Weder können Symbolleisten durch Hinzufügen oder Entfernen von Buttons individuell gestaltet werden, noch können Menüpunkte in ähnlicher Weise strukturell verändert werden, mit Ausnahme des Menüs *Tools* (siehe *Usability*, S. 150).

Makros können nicht aufgezeichnet werden, es besteht jedoch durch die VBA-Anbindung eine mächtige Möglichkeit, wiederkehrende Aufgaben effizient und benutzerfreundlich zu behandeln.

6 Empirische Bewertung der Werkzeug-Bedienung

In diesem Kapitel wird analysiert, wie schnell und effizient ein ausgewähltes Projekt mit Hilfe der getesteten Softwareentwicklungswerkzeuge erstellt werden kann. Hierbei wird einerseits die Modellierung anhand verschiedener UML-Diagrammtypen betrachtet, andererseits wird die Funktionalität im Zusammenhang mit Round-Trip Engineering (Codegenerierung und Rückführen von Quellcode in das jeweilige Modell) analysiert.

6.1 Zugrundeliegendes Projekt - 'Bestellverwaltung'

Angenommen wurde die Erstellung eines Softwarepakets zur Abwicklung von Bestellungen eines Unternehmens, das im Bereich des Internet-Versandhandels von verschiedenen Getränken tätig ist.

6.1.1 Natürlichsprachliche Spezifikation

Das Unternehmen hält verschiedene Produkte (Getränkesorten) auf Lager, die von Kunden bestellt werden können. Jedes Produkt wird unter einer eindeutigen Produktnummer geführt, weiters sind der Hersteller und die Bezeichnung des Produkts bekannt. Für die einzelnen Produkttypen (Wein, Bier etc.) können noch jeweils verschiedene Angaben gemacht werden, etwa der Jahrgang eines Weins, oder die Stammwürze einer Sorte Bier.

Die Kunden der Firma werden durch eine eindeutige Kundennummer identifiziert, weiters werden Name und Adresse des Kunden im System gespeichert.

Ein Kunde kann verschiedene Bestellungen aufgeben. Im Zuge einer Bestellung gibt der Kunde an, welche Produkte er in welcher Menge beziehen möchte. Außerdem kann er einen präferierten Liefertermin angeben.

6.1.2 Use-Case Modell

Es werden drei Anwendungsfälle unterschieden:

Use-Case	Beschreibung
Bestellungen verwalten	Das Bestellverwaltungssystem ist dafür verantwortlich, daß sämtliche Bestellungen korrekt abgewikkelt werden. Es prüft bei neuen Bestellungen, ob die Produkte in der gewünschten Menge vorrätig sind und wenn ja, fügt es diese zur aktuellen Bestellung hinzu.
Lagerstand aktualisieren	Es erfolgt eine Prüfung, ob das gewünschte Produkt in der erforderlichen Menge lagernd ist. Wenn die Bestellung durchgeführt wird, muß der Lagerstand aktualisiert werden, d.h. der aktuelle Lagerstand des entsprechenden Produkts wird um die bestellte Menge verringert.
Bestellung aufgeben	Ein Kunde gibt eine Bestellung auf. Er gibt an, welche Produkte er in welcher Menge erwerben möchte.

6.2 Beschreibung der Analyse

6.2.1 Vorgangsweise

Um allgemeingültige, empirisch nachvollziehbare Aussagen über die Performanz eines Tools treffen zu können, werden die einzelnen Aktionen (Maus, Tastatur) für die verschiedenen Schritte beim Anlegen eines bestimmten Diagrammtyps gezählt. Im Punkt *Round-Trip Engineering*, ab S. 235 wird die Analyse anhand einer Zeitmessung durchgeführt.

6.2.2 Erläuterung der tabellarischen Bewertung

Die Auswertungstabellen sind folgendermaßen aufgebaut:

- Spalte 1:
  Auflistung der einzelnen Modellierungsschritte
- Spalte 2:
  Erläuterung der Vorgangsweise der Modellierung
- Tabellen in der Dokumentmarginale:
  Anzahl der jeweils benötigte Aktionen

Hierbei wird unterschieden zwischen

- *Maus*
  Mausklick (einfacher, bzw. Doppelklick), auch zusammen mit Drag&Drop
- *Tastatur*
  Tastatureingabe, etwa zur Eingabe eines Element-Bezeichners

Für sämtliche Einzelaktionen wurde als Wertung "1" angenommen.

Verwendete Abkürzungen:

- LK Klick mit der linken Maustaste
- RK Klick mit der rechten Maustaste
- DK Doppelklick mit der linken Maustaste

6.2.3 Einschränkungen

Bei der Interpretation der Ergebnisse sind einige Nebenbedingungen zu beachten:

- Es wird von einem bereits "auf dem Papier" bestehenden Modell ausgegangen, woraufhin versucht wird, dieses Modell möglichst effizient im entsprechenden Tool einzutragen. Diese Art der Erstellung entspricht also nur in bestimmten Bereichen einem realen Systementwicklungsprozeß, bei dem etwa im Fall des Klassendiagramms nach und nach die verschiedenen Klassen zusammen mit einigen Attributen oder Operationen angelegt werden und erst in weiteren Schritten die bestehenden Informationen verfeinert werden (zusätzliche Klassen etc.).
- Das zugrundeliegende Modell beschreibt nur einen Ausschnitt des realen Problembereichs. Jedoch erscheint diese Einschränkung in folgender Hinsicht akzeptabel:
  - Die einzelnen Diagramme sollten für diese Art der Auswertung nicht übermäßig komplex sein.
  - Bestimmte Tools sind im Umfang ihrer UML-Darstellungsmöglichkeiten eingeschränkt (siehe Kapitel 3). Somit wurde versucht, einen "kleinsten gemeinsamen Nenner" der Diagrammelemente zu finden, damit alle Tools gleichwertig getestet werden können.
- Um die empirischen Tests nicht unnötig kompliziert zu gestalten, auch in Hinsicht auf die Analyse der Ergebnisse, wurden die einzelnen Aktionen (Maus, Tastatur) in ihrer Wertigkeit gleichgesetzt. Natürlich ist anzumerken, daß die konkrete Geschwindigkeit beim Erstellen eines Diagramms weitgehend auch davon abhängt, wie groß etwa die einzelnen Mauswege sind, oder wie effizient strukturiert die einzelnen Menüs und Dialogfenster von einem bestimmten Tool angeboten werden. Auch müssen in diesem Zusammenhang im realen Arbeitsprozeß sicherlich Präferenzen des einzelnen Anwenders berücksichtigt werden.
- Außerdem ist bei der Erstellung eines Klassendiagramms zu beachten, daß die Vorgangsweise in dieser Auswertung darin besteht, die benötigten Elemente zuerst graphisch anzulegen und daraufhin die notwendigen Informationen (wie Attribute, Operationen etc.) einzutra-

gen. Es besteht bei den meisten Tools auch die Möglichkeit, Klassen u.a. direkt im Data Dictionary anzulegen und nachfolgend das Klassendiagramm mit den gewünschten Elementen zu füllen. Jedoch wurde diese Vorgangsweise nicht gewählt, da sie zwar in bestimmten Fällen schneller von statten geht, jedoch weniger intuitiv erscheint.

6.3 Erstellung des Klassendiagramms

6.3.1 Zusammenfassung der Ergebnisse

In Abb. 6–1 können die Ergebnisse der Analyse der Klassendiagrammerstellung abgelesen werden. Auffallend ist, daß der "Testsieger" Paradigm Plus nur etwas mehr als halb so viele Aktionen benötigt, wie das "Schlußlicht" OTW-2.

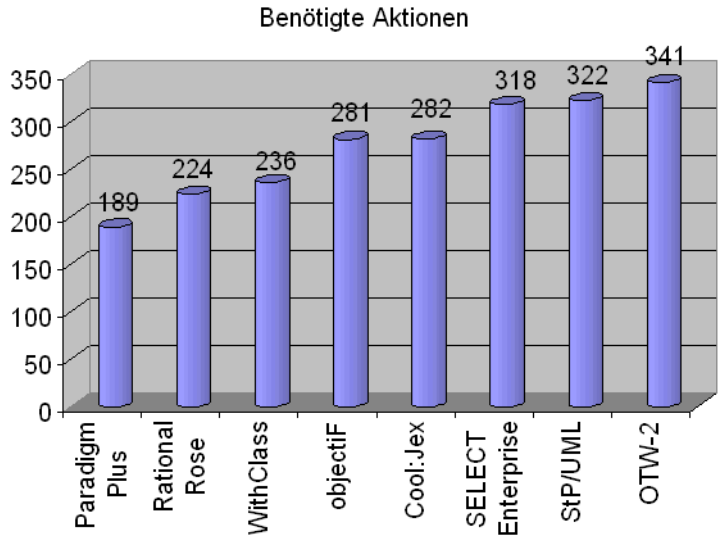

Abb. 6–1
Ergebnisse der Klassendiagrammerstellung

6.3.2 Interpretation der Ergebnisse

Generell können die Programme in zwei Gruppen geteilt werden. Die erste Gruppe erlaubt die Spezifikation aller relevanten Parameter der Modellelemente in einem String, der direkt in das Diagramm eingetragen werden kann. Bei den Programmen der zweiten Gruppe können die Eigenschaften der Elemente nur über Dialoge angegeben oder verändert werden. Programme der ersten Gruppe (Paradigm Plus, Rational Rose, Cool:Jex) schnitten bei der Bewertung deutlich besser ab, als die übrigen Werkzeuge.

Cool:Jex erhielt zwar nur eine mittelmäßige Bewertung, jedoch muß dabei berücksichtigt werden, daß bei diesem Programm zusätzlich recht hoher Aufwand durch das Vorbereiten der benötigten Stereotypen entstand.

Das gute Abschneiden von WithClass ist vor allem darauf zurückzuführen, daß die Relationen sehr einfach anzulegen sind. Die gewünschte Multiplizität kann über die Werkzeugleiste eingestellt werden, außerdem fällt die Eingabe der Rollenbezeichnungen weg.

Der Sieger in dieser Disziplin, Paradigm Plus, stellt Werkzeuge bereit, mit denen auf einfache Art Attribute und Methoden in eine Klasse eingefügt werden können.

Bei SELECT Enterprise und OTW-2 sind die schlechten Ergebnisse in erster Linie auf die etwas zeitaufwendige Bedienung der komplexen Dialoge zurückzuführen. StP/UML wiederum besitzt einen etwas umständlichen Mechanismus für das Anlegen von Relationen.

6.3.3 Klassendiagramm - Cool:Jex

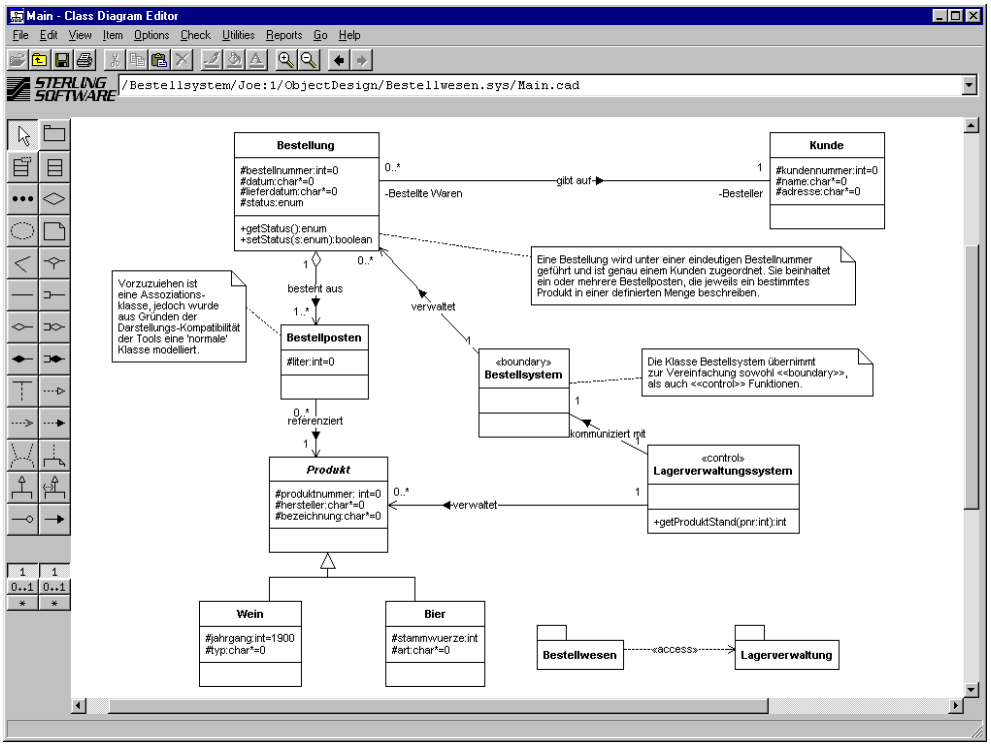

Abb. 6–2
Klassendiagramm in Cool:Jex

Schritt	Vorgangsweise
01 Anlegen der Pakete	Markieren der Phasenversion im Browser, danach Auswahl des Menübefehls *File - New - Package Version*. Eingabe der Bezeichnung für das Paket und Bestätigung mit der Enter-Taste. Das Anlegen des zweiten Pakete läuft ebenso ab.

Benötigte Aktionen

Maus:	*5*
Tastatur:	*2*
	7

Schritt	Vorgangsweise	Benötigte Aktionen
02 Anlegen eines neuen Klassendiagramms	Markieren des Paketes *Bestellwesen* im Browser, danach Auswahl des Menübefehles *File - New - Class Diagram*. Eingabe der Bezeichnung für die Klasse und Bestätigung mit der Enter-Taste.	*Maus: 3* *Tastatur: 1* *4*
03 Anlegen eines weiteren Klassendiagramms im zweiten Paket	Um die benötigten Klassen im zweiten Paket (Lagerverwaltung) anlegen zu können, muß auch in diesem ein Klassendiagramm angelegt werden (Diagramm2). Dies geschieht wie bereits in Punkt 02 beschrieben.	*Maus: 3* *Tastatur: 1* *4*
04 Anlegen der Klassen Kunde, Bestellung, Bestellposten und Bestellsystem im Paket Bestellwesen	LK auf das Klassensymbol in der Werkzeugleiste. Anlegen einer Klasse durch LK auf den Zeichenbereich, danach Eingabe des Klassennamens und Bestätigung mit der Enter-Taste. Am Ende der Aktion Auswahl des Werkzeugs *Select* aus der Werkzeugleiste.	*Maus: 6* *Tastatur: 4* *10*
05 Anlegen der Klassen Produkt, Wein, Bier und Lagerverwaltungssystem im Paket Lagerverwaltung	Analog Punkt 04	*Maus: 6* *Tastatur: 4* *10*
06 Anlegen der Attribute der Klasse Kunde	Durch LK auf den Attribut-Bereich der Klasse im Klassendiagramm wird der Cursor an die entsprechende Stelle gesetzt. Alle Attribut-Parameter werden über den hier einzugebenden Textstring gesetzt. Dieser wird daher als bis zu vier Bezeichnungsaktionen gerechnet (je nachdem, ob der Initialwert gesetzt wird, oder nicht).	*Maus: 1* *Tastatur: 19* *20*
07 Anlegen der Attribute der Klasse Bestellung	Analog Punkt 06	*Maus: 1* *Tastatur: 19* *20*
08 Anlegen der Attribute der Klasse Bestellposten	Analog Punkt 06	*Maus: 1* *Tastatur: 4* *5*
09 Anlegen der Attribute der Klasse Produkt	Analog Punkt 06	*Maus: 1* *Tastatur: 16* *17*

Schritt	Vorgangsweise	Benötigte Aktionen
10 Anlegen der Attribute der Klasse Wein	Analog Punkt 06	*Maus:* 1 *Tastatur:* 8 9
11 Anlegen der Attribute der Klasse Bier	Analog Punkt 06	*Maus:* 1 *Tastatur:* 7 8
12 Anlegen des Stereotyps *control* und Vorbereiten des bereits vorhandenen Stereotyps *boundary*	Auswahl des Menübefehls *Utilities - Customize* aus dem Hauptmenü des Browsers. Auswahl des Eintrages *Miscellaneous Customization* im folgenden Dialog, danach LK auf die Schaltfläche *Next*. Auswahl von *Stereotype Customization* im nächsten Dialog, danach LK auf *Next*. In den nächsten beiden Dialogen wird zunächst *location*, dann *corporate* ausgewählt. Eigentlich wird das Stereotyp gar nicht angelegt, sondern nur der Kombination von Notationselement und Diagramm zur Verfügung gestellt. Im nun angezeigten *Stereotype Availability Editor* wird nun dafür gesorgt, daß Klassen im Klassendiagramm Zugriff auf das Stereotyp *control* haben. Hier muß auch das, an sich schon vorhandene, Stereotyp *boundary* für die Verwendung im Klassendiagramm vorbereitet werden.	*Maus:* 35 *Tastatur:* 1 36
13 Klasse Lagerverwaltungssystem erhält Stereotyp *control*	RK auf die Klasse im Diagrammeditor, dann Auswahl von *Edit Properties* aus dem Kontextmenü. Auswahl von *control* aus dem Kombinationsfeld *Stereotyp*.	*Maus:* 5 *Tastatur:* 0 5
14 Klasse Bestellsystem erhält Stereotyp *boundary*	Analog Punkt 13	*Maus:* 5 *Tastatur:* 0 5
15 Klassentyp von Klasse Produkt wird auf *abstrakt* gesetzt	RK auf die Klasse im Diagrammeditor, dann Auswahl von *Edit Properties* aus dem Kontextmenü. Markieren der Checkbox *IsAbstract*, danach mit dem OK-Button bestätigen.	*Maus:* 4 *Tastatur:* 0 4
16 Anlegen der beiden Methoden der Klasse Bestellung	Durch LK auf den Methoden-Bereich der Klasse im Klassendiagramm, wird der Cursor an die entsprechende Stelle gesetzt. Alle Parameter der Methode werden über den hier einzugebenden Textstring gesetzt.	*Maus:* 1 *Tastatur:* 7 8

Schritt	Vorgangsweise	Benötigte Aktionen
17 Anlegen der Methode der Klasse Lagerverwaltungssystem	Analog Punkt 16	*Maus: 1* *Tastatur: 4* *5*
18 Einzeichnen der Klassen des Paketes Lagerverwaltung im Klassendiagramm des Paketes Bestellwesen	LK auf das Klassensymbol in der Werkzeugleiste. Anlegen einer Klasse durch LK auf den Zeichenbereich, danach RK auf die neu angelegte Klasse und Auswahl von *Select Classname* aus dem Kontextmenü. Im folgenden Dialog kann eine bereits vorhandene Klasse ausgewählt werden. Druck auf die Taste Paste, wodurch der Name der gewählten Klasse in das Diagramm eingefügt und der Dialog geschlossen wird. Die anderen Klassen wie oben beschrieben in das Diagramm einfügen, danach das Werkzeug *Select* aus der Werkzeugleiste selektieren.	*Maus: 22* *Tastatur: 0* *22*
19 Anlegen der Generalisation zwischen Produkt und Wein, bzw. Produkt und Bier	LK auf das Symbol *Generalization Connector* in der Werkzeugleiste. LK auf die Klasse Produkt, danach auf die Klasse Wein, um Wein von Produkt abzuleiten. Gleiche Aktion für die Klassen Produkt und Bier.	*Maus: 5* *Tastatur: 0* *5*
20 Anlegen der Assoziation zwischen Produkt und Bestellposten	LK auf das Symbol *Association* in der Werkzeugleiste, danach LK zuerst auf Bestellposten, danach auf Produkt. Eingabe der Bezeichnung der Assoziation, anschließend LK auf das Multiplizitäts-Symbol auf der Seite der Klasse Bestellposten und ersetzen der 1 durch 0..*. RK auf die Kante und aufrufen des Eigenschaftsdialoges über das Kontextmenü. Festlegen der Navigierbarkeit auf der ersten Seite des Dialoges, bestätigen mit *ok*.	*Maus: 9* *Tastatur: 2* *11*
21 Anlegen der Assoziation zwischen Produkt und Lagerverwaltungssystem	Analog Punkt 20	*Maus: 8* *Tastatur: 2* *10*
22 Anlegen der Assoziation zwischen Kunde und Bestellung	Analog Punkt 20, jedoch zusätzliche Angabe von Rollennamen auf beiden Seiten der Aggregation. Hierzu LK unterhalb jedes Endes der Beziehung und Eingabe des Rollennamens.	*Maus: 5* *Tastatur: 4* *9*

Schritt	Vorgangsweise	Benötigte Aktionen
23 Anlegen der Assoziation zwischen Bestellsystem und Lagerverwaltungssystem und Bestellung	Analog Punkt 20	*Maus:* 3 *Tastatur:* 0 3
24 Anlegen der Assoziation zwischen Bestellung und Bestellsystem	Analog Punkt 20	*Maus:* 8 *Tastatur:* 2 10
25 Anlegen der Aggregation zwischen Bestellposten und Bestellung	Auswählen des *Aggregation*-Werkzeugs aus der Werkzeugleiste, danach LK auf die Klasse Bestellung und Bestellposten. Eingabe des Bezeichners für die Aggregation und setzen der Multiplizität auf der Seite der Klasse Bestellposten.	*Maus:* 9 *Tastatur:* 2 11
26 Anlegen der Notiz samt Verbindung mit der Klasse Bestellung	Auswählen des *Note*-Werkzeugs aus der Werkzeugleiste, anlegen des Textfeldes im Diagramm und Eingabe des Textes. Auswählen des *Note Connector*-Werkzeugs und verbinden der Klasse Bestellung mit der Notiz.	*Maus:* 5 *Tastatur:* 1 6
27 Anlegen der beiden anderen Notizen	Auswählen des *Note*-Werkzeugs aus der Werkzeugleiste, anlegen des Textfeldes im Diagramm und Eingabe des Textes. Auswählen des *Note Connector*-Werkzeugs und verbinden der Klasse Bestellung mit der Notiz.	*Maus:* 10 *Tastatur:* 2 12
28 Anlegen der Pakkage-Symbole	Auswählen des *Package*-Werkzeugs aus der Werkzeugleiste, anlegen des Pakets im Diagramm und Eingabe des Bezeichners. Ebenso für das zweite Package.	*Maus:* 3 *Tastatur:* 2 5
29 Anlegen der Abhängigkeit zwischen den beiden Paketen	Auswählen des *Dependency*-Werkzeugs aus der Werkzeugleiste, LK auf Bestellwesen, danach auf Lagerverwaltung. RK auf die Abhängigkeit und Auswahl von *Edit Properties* aus dem Kontextmenü. Festlegen des Stereotyps *access*, bestätigen mit OK.	*Maus:* 8 *Tastatur:* 0 8
Gesamt		***282***

6.3.4 Klassendiagramm - objectiF

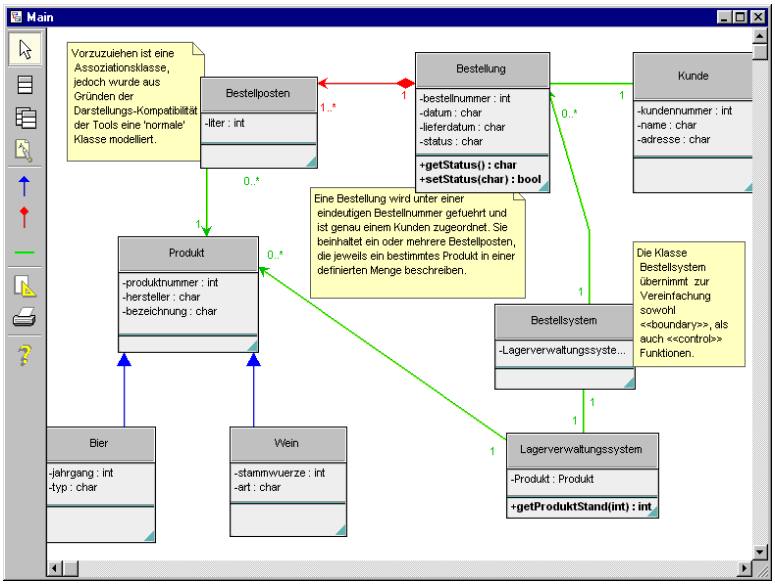

Abb. 6–3
Klassendiagramm in objectiF

Schritt	Vorgangsweise	*Benötigte Aktionen*
01 Anlegen der Pakete	RK im Package-Browser auf den Knoten Package, danach Auswahl des Befehls *Package anlegen* aus dem Kontextmenü. Es wird ein Dialog geöffnet, in dem der Bezeichner des Paketes eingegeben wird (Cursor schon an der richtigen Stelle, mit Enter bestätigen). Gleiche Aktion für das Paket Lagerverwaltung.	*Maus: 4* *Tastatur: 2* *8*
02 Anlegen eines neuen Klassendiagramms	RK im Package-Browser auf den Eintrag *Klassendiagramme*, danach Auswahl des Befehls *Klassendiagramm anlegen* aus dem Kontextmenü. Es wird ein Dialog geöffnet, in dem der Bezeichner des Paketes eingegeben wird (Cursor schon an der richtigen Stelle, mit Enter bestätigen).	*Maus: 2* *Tastatur: 1* *3*
03 Öffnen des Klassendiagramms	DK auf den Eintrag für das Klassendiagramm im Browser	*Maus: 1* *Tastatur: 0* *1*
04 Anlegen der Klasse Kunde	LK in der Werkzeugleiste auf das Klassen-Symbol. Anlegen der Klasse und Eingabe ihres Bezeichners. Danach markieren der Checkbox *öffentlich* und bestätigen der Aktion mit OK.	*Maus: 3* *Tastatur: 1* *4*
05 Anlegen der Klasse Bestellung	Analog Punkt 04	*Maus: 3* *Tastatur: 1* *4*

Schritt	Vorgangsweise	Benötigte Aktionen
06 Anlegen der Klasse Bestellposten	Analog Punkt 04	*Maus:* 3 *Tastatur:* 1 4
07 Anlegen der Klasse Produkt	Analog Punkt 04, jedoch muß zusätzlich die Checkbox *abstrakt* markiert werden.	*Maus:* 4 *Tastatur:* 1 5
08 Anlegen der Klasse Bier	Analog Punkt 04	*Maus:* 3 *Tastatur:* 1 4
09 Anlegen der Klasse Wein	Analog Punkt 04	*Maus:* 3 *Tastatur:* 1 4
10 Anlegen der Stereotypen *control* und *boundary*	Jetzt werden die beiden noch nicht vorhandenen Stereotypen angelegt, die beim Anlegen der nächsten beiden Klassen benötigt werden. Auswahl des Befehls *Stereotypen* aus dem Hauptmenü, es wird ein Fenster angelegt in dem die bestehende Stereotypen-Hierarchie dargestellt ist. Öffnen des Knotens *Type*, RK auf den Knoten *Class* und Auswahl von *anlegen - Stereotyp* aus dem Kontextmenü. Im folgenden Dialog wird der Bezeichner des Stereotyps eingegeben und mit der Enter-Taste bestätigt. Auf die selbe Weise wird das zweite Stereotyp angelegt und danach das Fenster mit dem Button *schließen* verlassen.	*Maus:* 8 *Tastatur:* 2 10
11 Anlegen der Klasse Bestellsystem	Analog Punkt 04, jedoch wird zusätzlich in dem Kombinationsfeld Stereotyp der Eintrag *boundary* ausgewählt, eines der beiden zuvor angelegten Stereotypen.	*Maus:* 5 *Tastatur:* 1 6
12 Anlegen der Klasse Lagerverwaltungssystem	Analog Punkt 04, jedoch wird zusätzlich in dem Kombinationsfeld Stereotyp der Eintrag *control* ausgewählt	*Maus:* 5 *Tastatur:* 1 6
13 Vergrößern der Klassensymbole	Vergrößern aller Klassensymbole im Diagramm, damit der Bereich für die Attribute sichtbar wird. Geschieht dies nicht, muß der Benutzer beim Anlegen jedes Attributes einen Befehl aus dem Kontextmenü wählen, statt dessen genügt jetzt ein RK auf den Attribut-Bereich einer Klasse.	*Maus:* 8 *Tastatur:* 0 8

Schritt	Vorgangsweise	Benötigte Aktionen
14 Anlegen der Attribute der Klasse Kunde	RK in den Attribut-Bereich der Klasse. Eingabe der Bezeichnung des Attributes. In einem weiteren Textfeld wird aus dem semantischen Namen automatisch eine Deklarationszeile generiert, in welcher der Typ des Attributes defaultmäßig auf *int* gesetzt wird. Soll nun der Typ geändert werden, muß zuerst der alte gelöscht und durch den neuen ersetzt werden. Eigentlich eine unnötige Behinderung, es wurden trotzdem nur zwei Aktionen gerechnet, da der zu ersetzende Typ mit einem DK gewählt und ersetzt werden kann. Der Initialwert des Attributes wird händisch in die Deklarationszeile eingetragen. Bestätigt wird der Vorgang mit *Anlegen* wodurch der Dialog nicht geschlossen wird und sofort ein neues Attribut angelegt werden kann, bzw. durch *Schließen*, nach dem Anlegen des letzten Attributes.	*Maus:* 10 *Tastatur:* 8 *18*
15 Anlegen der Attribute der Klasse Bestellung	Analog Punkt 14, bei der Angabe des Datentyps des Attributes *status* wird dieser aus dem Typenfenster im unteren Bereich des Dialoges ausgewählt.	*Maus:* 14 *Tastatur:* 9 *23*
16 Anlegen der Attribute der Klasse Bestellposten	Analog Punkt 14	*Maus:* 4 *Tastatur:* 2 *6*
17 Anlegen der Attribute der Klasse Produkt	Analog Punkt 14	*Maus:* 11 *Tastatur:* 10 *21*
18 Anlegen der Attribute der Klasse Wein	Analog Punkt 14	*Maus:* 7 *Tastatur:* 5 *12*
19 Anlegen der Attribute der Klasse Bier	Analog Punkt 14	*Maus:* 6 *Tastatur:* 4 *10*

Schritt	Vorgangsweise	Benötigte Aktionen
20 Anlegen der beiden Methoden der Klasse Bestellung	Öffnen des Dialoges zum Anlegen von Methoden durch RK in den Methoden-Bereich des Klassensymbols im Diagramm. Eingabe des Bezeichners, Markieren des voreingestellten Typs *void* in der Deklarationszeile und auswählen des Typs *Statustyp* aus der Typhierarchie im unteren Teil des Dialoges. Bestätigen der Eingabe durch den Button *Anlegen*, und wiederholen der Aktion für die zweite Methode der Klasse. Diesmal muß sowohl der default-Typ *void* durch *bool* ersetzt werden, als auch ein Parameter vom Typ *Statustyp* angelegt werden. Dieser wird nach dem C++ Syntax in der Deklarationszeile in die Klammern nach dem Methodennamen plaziert.	*Maus:* 11 *Tastatur:* 3 14
21 Anlegen der Methode der Klasse Lagerverwaltungssystem	Analog Punkt 20	*Maus:* 5 *Tastatur:* 4 9
22 Anlegen der Generalisation zwischen Produkt und Wein, bzw. Produkt und Bier	LK auf das Symbol *Generalisierung* in der Werkzeugleiste. LK auf die Klasse Produkt, danach auf die Klasse Wein, um Wein von Produkt abzuleiten. Gleiche Aktion für die Klassen Produkt und Bier.	*Maus:* 5 *Tastatur:* 0 5
23 Anlegen der Assoziation zwischen Produkt und Bestellposten	LK auf das Symbol *Assoziation* in der Werkzeugleiste, danach LK zuerst auf Bestellposten, danach auf Produkt. RK auf die Kante und öffnen des Eigenschaftsdialoges über das Kontextmenü. Eingabe der Multiplizität und, auf der zweiten Dialogseite, der Navigierbarkeit. Bestätigung der Aktion mit der Enter-Taste.	*Maus:* 9 *Tastatur:* 1 10
24 Anlegen der Assoziation zwischen Produkt und Lagerverwaltungssystem	Analog Punkt 23	*Maus:* 10 *Tastatur:* 1 11
25 Anlegen der Assoziation zwischen Kunde und Bestellung	Analog Punkt 23, jedoch werden im Eigenschaftendialog zusätzlich Rollennamen vergeben, und die Assoziation wird beidseitig navigierbar gemacht.	*Maus:* 14 *Tastatur:* 3 17

Schritt	Vorgangsweise	*Benötigte Aktionen*
26 Anlegen der Assoziation zwischen Bestellsystem und Lagerverwaltungssystem	Analog Punkt 23	*Maus: 10* *Tastatur: 2* *12*
27 Anlegen der Assoziation zwischen Bestellung und Bestellsystem	Analog Punkt 23	*Maus: 9* *Tastatur: 2* *11*
28 Anlegen der Aggregation zwischen Bestellposten und Bestellung	Analog Punkt 23, jedoch wird statt dem Werkzeug zum Anlegen von Assoziationen, das *Aggregation*-Werkzeugs aus der Werkzeugleiste aktiviert.	*Maus: 10* *Tastatur: 1* *11*
29 Anlegen der Notiz samt Verbindung mit der Klasse Bestellung	Auswählen des *Note*-Werkzeugs aus der Werkzeugleiste und anlegen des Textfeldes im Diagramm. DK auf das neue Textfeld um den Cursor zu erhalten und Eingabe des Textes. Die Notiz kann nicht mit einem Modellierungselement verbunden werden.	*Maus: 5* *Tastatur: 1* *6*
30 Anlegen der beiden anderen Notizen	Analog Punkt 29	*Maus: 10* *Tastatur: 2* *12*
31 Zuweisen der Klassen zu den zu Beginn generierten Packages	Verschieben der Klassen im Package-Browser per Drag&Drop in das jeweilige Paket.	*Maus: 8* *Tastatur: 0* *8*
32 Packages	Es besteht keine Möglichkeit Packages in objectiF grafisch darzustellen.	*Maus: 0* *Tastatur: 0* *0*
Gesamt		***281***

6.3.5 Klassendiagramm - OTW-2

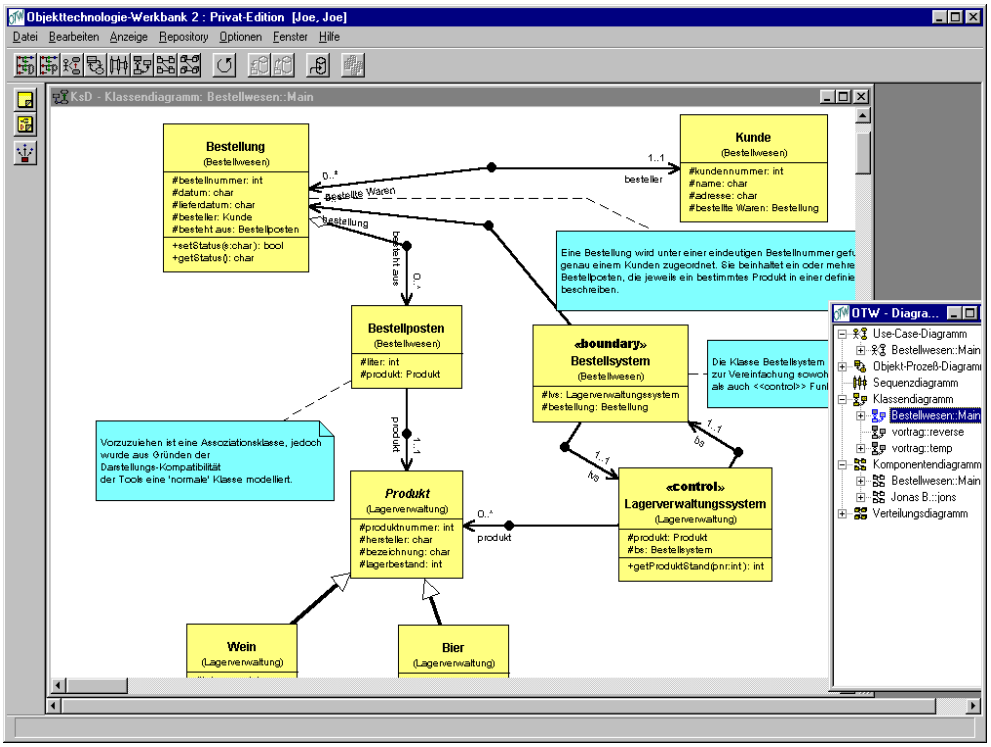

Abb. 6–4
Klassendiagramm in OTW-2

Schritt	Vorgangsweise	*Benötigte Aktionen*
01 Anlegen der Pakete	RK im Paketübersichtsbrowser auf den Knoten Paket, danach Auswahl des Befehls Neues Paket aus dem Kontextmenü. Es wird ein Dialog geöffnet, in dem der Bezeichner des Paketes (Bestellwesen) eingegeben wird (Cursor schon an der richtigen Stelle, mit Enter bestätigen). Gleiche Aktion für das Paket Lagerverwaltung.	*Maus:* 4 *Tastatur:* 2 6
02 Anlegen eines neuen Klassendiagramms	RK im Diagramm-Browser auf den Eintrag Klassendiagramm, danach Auswahl des Befehls Neues Diagramm aus dem Kontextmenü. Im folgenden Dialog muß aus einer Listbox eines der vorhandenen Pakete (Bestellwesen) ausgewählt werden. Danach wird mittels eines LK der Cursor auf das Textfeld für die Namenseingabe gesetzt und der Bezeichner des Diagramms eingegeben. Danach wird die Aktion mit der Enter-Taste bestätigt.	*Maus:* 4 *Tastatur:* 1 5
03 Anlegen der Klasse Kunde	LK in der Werkzeugleiste auf das Klassen-Symbol. Anlegen der Klasse und Eingabe ihres Bezeichners. Die Auswahl des richtigen Paketes im Eingabedialog entfällt, da das aktuelle Paket immer voreingestellt ist. Daher befindet sich der Cursor auch schon im Namensfeld. Aktion wird mit der Enter-Taste bestätigt.	*Maus:* 2 *Tastatur:* 1 3

Schritt	Vorgangsweise	Benötigte Aktionen
04 Anlegen der Klasse Bestellung	Wie bei Punkt 03 mit dem Unterschied, daß das Klassen-Werkzeug bereits ausgewählt ist.	*Maus: 1* *Tastatur: 1* *2*
05 Anlegen der Klasse Bestellposten	Wie bei Punkt 04	*Maus: 1* *Tastatur: 1* *2*
06 Anlegen der Klasse Produkt	Wie bei Punkt 04, jedoch muß zusätzlich im Eingabedialog das Paket Lagerverwaltung gewählt und danach der Cursor auf das Namensfeld gesetzt werden. Danach wird der Reiter Eigenschaften angewählt und dort der Klassentyp auf abstrakt gesetzt. Zuletzt wird die Aktion mit einem LK auf den OK-Button bestätigt.	*Maus: 6* *Tastatur: 1* *7*
07 Anlegen der Klasse Wein	Analog Punkt 04, jedoch mit vorheriger Auswahl des Paketes und Setzen des Cursors	*Maus: 3* *Tastatur: 1* *4*
08 Anlegen der Klasse Bier	Analog Punkt 07	*Maus: 3* *Tastatur: 1* *4*
09 Anlegen der Klasse Bestellsystem	LK auf die Zeichenfläche um Klasse anzulegen, Eingabe des Bezeichners im Eigenschaftendialog. LK auf den Reiter Stereotyp um das Stereotyp *boundary* anzulegen. LK auf die Schaltfläche *Stereotyp neuanlegen*, Eingabe der Bezeichnung und Bestätigung mit der Enter-Taste. Dialog verlassen mit LK auf OK-Taste.	*Maus: 6* *Tastatur: 2* *8*
10 Anlegen der Klasse Lagerverwaltungssystem	Analog Punkt 09, mit anlegen des Stereotyps *control*, jedoch mit vorheriger Auswahl des Paketes und setzen des Cursors. Da alle Klassen angelegt sind, wird in der Werkzeugleiste das *Verschieben/Auswählen*-Tool aktiviert.	*Maus: 9* *Tastatur: 2* *11*

Schritt	Vorgangsweise	Benötigte Aktionen
11 Anlegen der Attribute Kundennummer, Name und Adresse in der Klasse Kunde	RK auf die Klasse im Klassendiagramm um über das Kontextmenü in den Dialog *Auswahl von Attributen* zu gelangen. Eingabe des Bezeichners des Attributes (Kundennummer). Auswahl des Eigenschaften Reiters, danach selektieren des Attributtyps aus dem Kombinationsfeld Typ. Einstellen des Variablentyps Wert. LK auf den Reiter *Erweiterte Eigenschaften* und Eingabe des Initialwertes für das Attribut. LK auf den Button Übernehmen, um die Eingabe zu bestätigen. LK auf den Auswahl-Reiter und Eingabe der beiden anderen Attribute wie oben beschrieben.	*Maus: 23* *Tastatur: 6* *29*
12 Anlegen der Attribute der Klasse Bestellung	Analog Punkt 11	*Maus: 29* *Tastatur: 7* *36*
13 Anlegen der Attribute der Klasse Bestellposten	Analog Punkt 11	*Maus: 9* *Tastatur: 2* *11*
14 Anlegen der Attribute der Klasse Produkt	Analog Punkt 11	*Maus: 30* *Tastatur: 8* *38*
15 Anlegen der Attribute der Klasse Wein	Analog Punkt 11	*Maus: 16* *Tastatur: 4* *20*
16 Anlegen der Attribute der Klasse Bier	Analog Punkt 11	*Maus: 14* *Tastatur: 3* *17*
17 Anlegen der Methoden der Klasse Bestellung	RK auf die Klasse in der Diagrammansicht und Auswahl des Eintrages *Prozesse* aus dem Kontextmenü. Eingabe des Methodennamens, Umschalten auf die Dialogseite *Parameter* und Auswahl des Rückgabetyps aus einer Droplist. Betätigen der Schaltfläche *Übernehmen* und wiederholen der Aktion für die zweite Methode. Hierbei muß zusätzlich ein Parameter spezifiziert werden. Dies geschieht ebenfalls auf der Dialogseite *Parameter*, durch Eingabe des Namens in das hierfür vorgesehene Textfeld, Betätigen der Schaltfläche *neu*, Markieren des neuen Parameters in der Listbox und Auswahl eines Datentyps.	*Maus: 17* *Tastatur: 2* *19*

Schritt	Vorgangsweise	*Benötigte Aktionen*
18 Anlegen der Methoden der Klasse Lagerverwaltungssystem	Analog Punkt 17	*Maus: 12* *Tastatur: 2* *14*
19 Anlegen der Generalisation zwischen Produkt und Wein, bzw. Produkt und Bier	LK auf das Symbol *Neue Klassenvererbung* in der Werkzeugleiste. LK auf die Klasse Wein, danach auf die Klasse Produkt, um Wein von Produkt abzuleiten. Bestätigen des folgenden Dialoges mit Enter. Gleiche Aktion für die Klassen Produkt und Bier.	*Maus: 7* *Tastatur: 0* *7*
20 Anlegen der Assoziation zwischen Produkt und Bestellposten	LK auf das Symbol *Neue Rolle* in der Werkzeugleiste, danach LK zuerst auf Bestellposten, danach auf Produkt. Eingabe der Bezeichnung des Attributes der Klasse Bestellposten, auf das sich die Assoziation bezieht, Wechseln der Dialogseite auf *Eigenschaften* und Ersetzen des Min-Eintrages der Multiplizität auf 1. Bestätigen der Aktion mit OK.	*Maus: 6* *Tastatur: 2* *8*
21 Anlegen der Assoziation zwischen Produkt und Lagerverwaltungssystem	Analog Punkt 20	*Maus: 5* *Tastatur: 1* *6*
22 Anlegen der Assoziation zwischen Kunde und Bestellung	Analog Punkt 20, jedoch müssen hier zwei Rollenbeziehungen angelegt werden, damit die Beziehung beidseitig navigierbar ist.	*Maus: 10* *Tastatur: 3* *13*
23 Anlegen der Aggregation zwischen Bestellposten und Bestellung	Analog Punkt 20, im Eigenschaften Dialog des der Beziehung zugrundeliegenden Attributes muß der Rollentyp auf *Aggregation* gesetzt werden.	*Maus: 6* *Tastatur: 1* *7*
24 Anlegen der Assoziation zwischen Bestellsystem und Lagerverwaltungssystem	Analog Punkt 22	*Maus: 10* *Tastatur: 2* *12*

Schritt	Vorgangsweise	Benötigte Aktionen
25 Anlegen der Assoziation zwischen Bestellung und Bestellsystem	Analog Punkt 20	*Maus:* 5 *Tastatur:* 1 6
26 Anlegen der Notiz samt Verbindung mit der Klasse Bestellung	Auswählen des *Neuer Kommentar*-Werkzeugs aus der Werkzeugleiste, anlegen des Textfeldes im Diagramm und Eingabe des Textes. Auswählen des *Abhängigkeit*-Werkzeugs und verbinden der Klasse Bestellung mit der Notiz.	*Maus:* 6 *Tastatur:* 1 7
27 Anlegen der beiden anderen Notizen	Analog Punkt 26	*Maus:* 12 *Tastatur:* 2 14
28 Anlegen eines neuen Komponenten-diagramms	Da es in OTW-2 nicht möglich ist, Pakete in Klassendiagramme zu zeichnen, muß für die gewünschte Darstellung der beiden Pakete ein Komponentendiagramm angelegt werden. RK im Diagramm-Browser auf den Eintrag Komponentendiagramm, danach Auswahl des Befehls Neues Diagramm aus dem Kontextmenü. Im folgenden Dialog muß aus einer Listbox eines der vorhandenen Pakete (Bestellwesen) ausgewählt werden. Danach wird mittels eines LK der Cursor auf das Textfeld für die Namenseingabe gesetzt und der Bezeichner des Diagramms eingegeben. Danach wird die Aktion mit der Enter-Taste bestätigt.	*Maus:* 5 *Tastatur:* 1 6
29 Anlegen der beiden Pakete im Komponentendiagramm	Auswählen des Werkzeugs *Neuer Kommentar* aus der Werkzeugleiste und anlegen des Paketes mittels LK auf die Zeichenfläche. Auswahl des gewünschten Paketes aus der Liste der vorhandenen Pakete und bestätigen der Aktion mit OK. Wiederholen des Vorganges für das zweite Paket.	*Maus:* 7 *Tastatur:* 0 7
30 Anlegen der Abhängigkeit zwischen den beiden Paketen	Auswahl des Werkzeugs *Neue Benutzt-Beziehung*, LK auf Bestellwesen und danach auf Lagerverwaltungssystem. RK auf die Beziehung und Aufruf des Eigenschaftendialoges. Anlegen des Stereotyps *access* wie bei Punkt 09 beschrieben und Auswahl desselben.	*Maus:* 11 *Tastatur:* 1 12
Gesamt		***341***

6.3.6 Klassendiagramm - Paradigm Plus

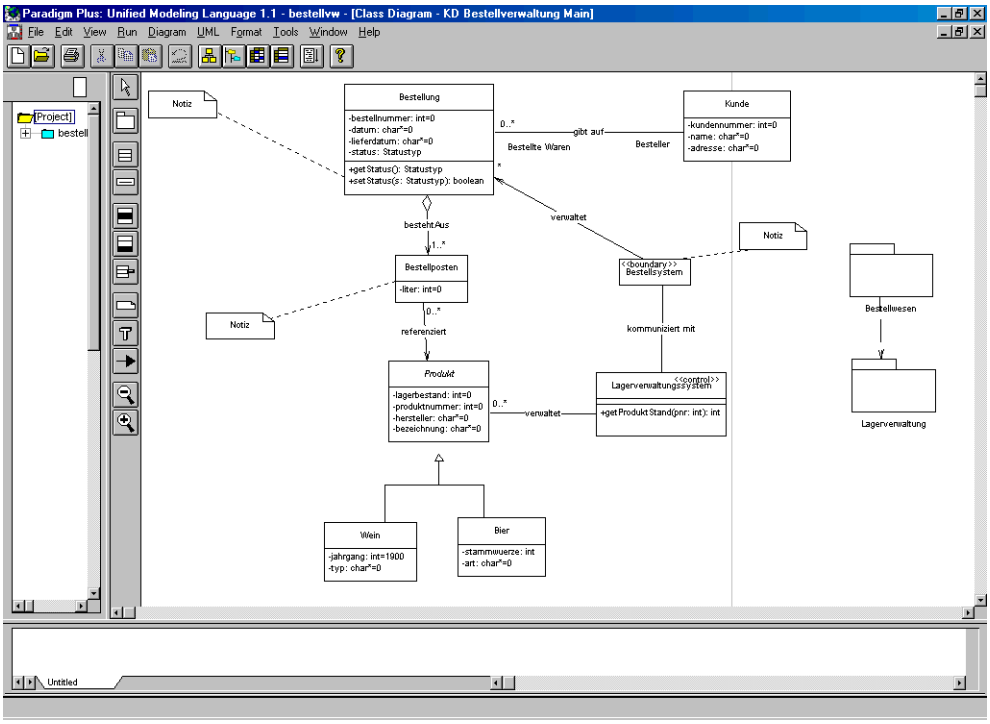

Abb. 6–5
Klassendiagramm in Paradigm Plus

Schritt	Vorgangsweise	Benötigte Aktionen	
01 Anlegen eines neuen Klassendiagramms	LK auf Menüleiste, Punkt *Diagram*, Auswahl des Unterpunktes *New...* Eingabe des Diagrammbezeichners im Dialogfenster und anschließend Bestätigen.	*Maus:* *Tastatur:* *4*	*3* *1*
02 Anlegen der Pakete	LK in der Werkzeugleiste auf das Component-Symbol. Durch zwei LK im Diagrammfenster werden die beiden Pakete angelegt. Abschließend LK auf Pfeilsymbol in der Werkzeugleiste, um Vorgang abzuschließen.	*Maus:* *Tastatur:* *4*	*4* *0*
03 Vergeben von Paket-Bezeichnern	DK auf das entsprechende Package. Im folgenden Dialogfenster wird der gewünschte Bezeichner eingetragen. und bestätigt.	*Maus:* *Tastatur:* *6*	*4* *2*
04 Anlegen der Klassen	LK auf das *Class*-Symbol in der Werkzeugleiste. Sämtliche benötigten Klassen werden im Diagrammfenster eingezeichnet. Durch Anwählen des Pfeilsymbols wird das wiederholte Einfügen neuer Klassen abgeschlossen.	*Maus:* *Tastatur:* *10*	*10* *0*

Schritt	Vorgangsweise	Benötigte Aktionen
05 Vergeben von Klassenbezeichnern, sowie Stereotypen	Vergeben der Bezeichner analog Punkt 03. Zusätzlich noch Vergabe von Stereotypen im selben Dialogfenster. Der Stereotyp *abstract* kann direkt ausgewählt werden, wobei die Stereotypen *control* und *boundary* als *User Defined Stereotypes* eingetragen werden müssen.	*Maus: 20* *Tastatur: 10* *30*
06 Anlegen der Attribute der Klasse Bestellung	LK auf Symbol *Attribute* in der Werkzeugleiste. LK in das Klassensymbol und ein neues Attribut wird angelegt. Im Textfeld kann sowohl der Bezeichner als auch der Typ, sowie der Initialwert direkt in der Form *name : typ = default* eingetragen werden. Dies kann ohne weitere Auswahl des Werkzeugleisten-Symbols für sämtliche Attribute durchgeführt werden.	*Maus: 5* *Tastatur: 4* *9*
07 Anlegen der Attribute der Klasse Kunde	Analog Punkt 06	*Maus: 3* *Tastatur: 3* *6*
08 Anlegen der Attribute der Klasse Bestellposten	Analog Punkt 06	*Maus: 4* *Tastatur: 4* *8*
10 Anlegen der Attribute der Klasse Wein	Analog Punkt 06	*Maus: 2* *Tastatur: 2* *4*
11 Anlegen der Attribute der Klasse Bier	Analog Punkt 06	*Maus: 2* *Tastatur: 2* *4*
12 Anlegen der Operationen der Klasse Bestellung	Analog Anlegen von Attributen (Punkt 06), Auswahl des Symbols *Operation* in der Werkzeugleiste. Eintrag der Operation in der Form *name (parameter: typ): return*	*Maus: 3* *Tastatur: 2* *5*
13 Eintragen der Operationen-Eigenschaften der Klasse Lagerverwaltungssystem	Analog Punkt 12	*Maus: 1* *Tastatur: 1* *2*

Schritt	Vorgangsweise	Benötigte Aktionen
14 Anlegen sämtlicher Beziehungen zwischen Klassen (Assoziation, Aggregation, Generalisation) und der Abhängigkeits-beziehung zwischen den Paketen	LK auf das Symbol *Relationship* in der Werkzeugleiste. LK in die Quell-Klasse, LK in die Ziel-Klasse. Im folgenden Dialogfenster Auswahl des Beziehungstyps (*Association*, *Aggregation* bzw. *Generalization*) - Beziehung wird angelegt. Bei Anlegen der zweiten Generalisation wird automatisch ein *Shared Target Style* der Beziehungen angewendet. Die Abhängigkeitsbeziehung zwischen den Paketen *Bestellwesen* und *Lagerverwaltung* muß nicht näher spezifiziert werden. Dies wird für sämtliche Beziehungen durchgeführt, danach Abschluß durch LK auf Pfeilsymbol in der Werkzeugleiste.	*Maus:* 31 *Tastatur:* 0 31
15 Eintragen der Eigenschaften der Assoziation *gibt auf* zwischen Kunde und Bestellung	DK auf die Assoziation. Im folgenden Dialogfenster, Eintrag des Bezeichners der Beziehung und der Rollennamen. Die Kardinalität von Bestellung muß ebenfalls geändert werden (wird mittels Dropdown-Menü ausgewählt).	*Maus:* 6 *Tastatur:* 3 9
16 Eintragen der Eigenschaften der Assoziation *referenziert* zwischen Bestellposten und Produkt	Analog Punkt 15, die Kardinalität muß geändert werden, Rollennamen werden keine vergeben, die Navigierbarkeit wird durch Auswahl des Punktes *Directed* eingestellt.	*Maus:* 7 *Tastatur:* 1 8
17 Eintragen der Eigenschaften der Assoziation *verwaltet* zwischen Lagerverwaltungssystem und Produkt	Analog Punkt 15, auch hier muß die Kardinalität von Produkt und die Navigierbarkeit geändert werden.	*Maus:* 4 *Tastatur:* 1 5
18 Eintragen der Eigenschaften der Assoziation *kommuniziert mit* zwischen Lagerverwaltungssystem und Bestellsystem	Analog Punkt 15	*Maus:* 4 *Tastatur:* 1 5

Schritt	Vorgangsweise	*Benötigte Aktionen*
19 Eintragen der Eigenschaften der Assoziation *verwaltet* zwischen Bestellsystem und Bestellung	Analog Punkt 15	*Maus: 2* *Tastatur: 1* *3*
20 Eintragen der Eigenschaften der Aggregation *bestehtAus* zwischen Bestellung und Bestellposten	Analog Punkt 15	*Maus: 5* *Tastatur: 1* *6*
21 Anlegen der Notizen	LK auf das Symbol *Comment* in der Werkzeugleiste. LK in das Klassendiagramm an den gewünschten Stellen. LK auf Pfeilsymbol in der Werkzeugleiste, um abzuschließen.	*Maus: 5* *Tastatur: 0* *5*
22 Eintragen der Notizinhalte	DK auf die Notiz, im Dialogfenster muß ein Bezeichner für die Notiz vergeben werden. Danach Auswahl der Karteikarte *Definition* und LK in das Textfeld. Hier kann der Inhalt der Notiz eingetragen und abschließend bestätigt werden.	*Maus: 12* *Tastatur: 6* *18*
23 Verbinden der Notizen mit den entsprechenden Klassen	LK auf das Symbol *Relationship* in der Werkzeugleiste. LK in die Klasse, LK in die Notiz - Verbindung wird angelegt. Abschluß mittels LK in das Pfeilsymbol in der Werkzeugleiste.	*Maus: 6* *Tastatur: 0* *6*
Gesamt		***189***

6.3.7 Klassendiagramm - Rational Rose

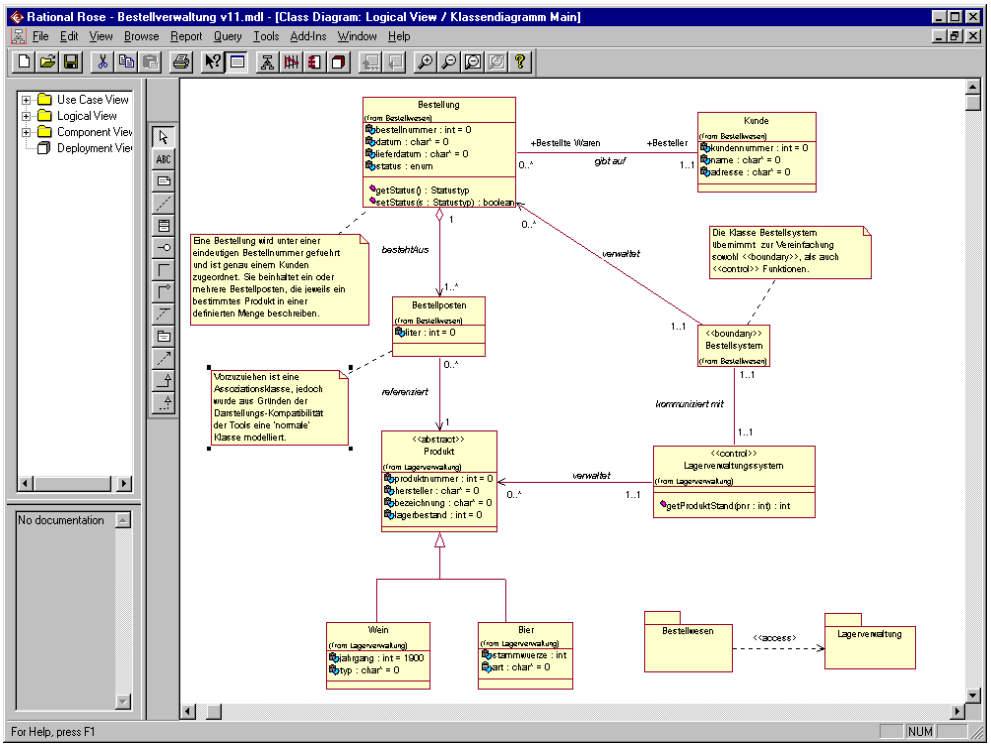

Abb. 6–6
Klassendiagramm in Rational Rose

Schritt	Vorgangsweise	*Benötigte Aktionen*
01 Anlegen eines neuen Klassendiagramms	RK im Browser auf den Eintrag Klassendiagramm, danach Auswahl des Befehls New - Class Diagram aus dem Kontextmenü. Danach Eingabe eines Bezeichners für das Diagramm und DK im Browser um es darzustellen.	*Maus: 4* *Tastatur: 1* *5*
02 Anlegen der Pakete	LK auf das Paketsymbol in der Werkzeugleiste. Jetzt mit gehaltener Shift-Taste und LK das Paket anlegen und einen Bezeichner eingeben. Das zweite Paket auf die gleiche Art anlegen. Die gedrückte Shift-Taste sorgt dafür, daß das Werkzeug nach der Aktion selektiert bleibt.	*Maus: 3* *Tastatur: 2* *5*
03 Anlegen aller Klassen	LK auf das Klassensymbol in der Werkzeugleiste. Jetzt mit gehaltener Shift-Taste und LK die Klasse anlegen und einen Bezeichner eingeben. Die übrigen Klassen auf die gleiche Art anlegen.	*Maus: 9* *Tastatur: 8* *17*
04 Zuweisen der Klassen zu Paketen	Im Browserfenster per Drag&Drop die Klassen auf die beiden Pakete aufteilen.	*Maus: 8* *Tastatur: 0* *8*

Schritt	Vorgangsweise	Benötigte Aktionen
05 Anlegen der Attribute der Klasse Kunde	RK auf die Klasse im Editor und auswählen des Befehls *New Attribute* aus dem Kontextmenü. Jetzt können alle Attribute textuell direkt in die Diagrammansicht geschrieben werden. Jede Zeile nimmt ein Attribut in der Form *name: typ = Initialwert* auf. Durch betätigen der Enter-Taste gelangt man in eine neue Zeile, d.h. es wird ein neues Attribut angelegt.	*Maus: 2* *Tastatur: 9* *11*
06 Anlegen der Attribute der Klasse Bestellung	Analog Punkt 05	*Maus: 2* *Tastatur: 11* *13*
07 Anlegen der Attribute der Klasse Bestellposten	Analog Punkt 05	*Maus: 2* *Tastatur: 3* *5*
08 Anlegen der Attribute der Klasse Produkt	Analog Punkt 05	*Maus: 2* *Tastatur: 12* *14*
09 Anlegen der Attribute der Klasse Wein	Analog Punkt 05	*Maus: 2* *Tastatur: 6* *8*
10 Anlegen der Attribute der Klasse Bier	Analog Punkt 05	*Maus: 2* *Tastatur: 5* *7*
11 Klasse Lagerverwaltungssystem erhält Stereotyp *control*	RK auf die Klasse im Diagramm und über das Kontextmenü den Eigenschaftendialog für die Klasse aufrufen (*Open Specification*). Auf der zu beginn aufgeschlagenen Dialogseite (*General*) wird im Kombinationsfeld Stereotyp die Zeichenkette *control* ausgewählt.	*Maus: 5* *Tastatur: 0* *5*
12 Klasse Lagerverwaltungssystem erhält Stereotyp *boundary*	Analog Punkt 11	*Maus: 2* *Tastatur: 3* *5*

Schritt	Vorgangsweise	Benötigte Aktionen
13 Klassentyp von Klasse Produkt auf *abstrakt* setzen	RK auf die Klasse im Diagramm und über das Kontextmenü den Eigenschaftendialog für die Klasse aufrufen. Mit einem LK auf den Reiter *Detail* die Dialogseite wechseln und dort die Checkbox *abstract* markieren.	*Maus: 5* *Tastatur: 0* *5*
14 Anlegen der beiden Methoden der Klasse Bestellung	Durch einen RK auf die Klasse im Klassendiagramm und Auswahl von *new Operation* aus dem Kontextmenü, wird der Cursor an die entsprechende Stelle gesetzt. Alle benötigten Parameter der Methode können werden über den hier einzugebenden Textstring gesetzt werden.	*Maus: 2* *Tastatur: 5* *7*
15 Anlegen der Methode der Klasse Lagerverwaltungssystem	Analog Punkt 14	*Maus: 2* *Tastatur: 3* *5*
16 Anlegen der Generalisation zwischen Produkt und Wein, bzw. Produkt und Bier	LK auf das Symbol *Generalization* in der Werkzeugleiste. LK mit gedrückter Shift-Taste auf die Klasse Wein, danach auf die Klasse Produkt, um Wein von Produkt abzuleiten. Gleiche Aktion für die Klassen Produkt und Bier. Die gedrückte Shift-Taste sorgt dafür, daß das Werkzeug nach der Aktion selektiert bleibt.	*Maus: 3* *Tastatur: 0* *3*
17 Anlegen der Assoziation zwischen Produkt und Bestellposten	LK auf das Symbol *Association* in der Werkzeugleiste, danach LK auf Bestellposten und ziehen mit gedrückter Maustaste nach Produkt. Eingabe der Bezeichnung der Assoziation, anschließend RK auf die Beziehung und Auswahl von *Open Specification* aus dem Kontextmenü. Einstellen der Multiplizität und Navigierbarkeit auf den Dialogseiten *Role A Detail* sowie *Role B Detail*.	*Maus: 10* *Tastatur: 3* *13*
18 Anlegen der Assoziation zwischen Produkt und Lagerverwaltungssystem	Analog Punkt 17	*Maus: 10* *Tastatur: 3* *13*
19 Anlegen der Assoziation zwischen Kunde und Bestellung	Analog Punkt 17, jedoch zusätzliche Angabe von Rollennamen auf beiden Seiten der Aggregation. Die Rollennamen werden auf der Dialogseite *Role A Detail* bzw. *Role B Detail* angegeben.	*Maus: 9* *Tastatur: 5* *14*

Schritt	Vorgangsweise	Benötigte Aktionen
20 Anlegen der Aggregation zwischen Bestellposten und Bestellung	Analog Punkt 17, jedoch wird statt dem Werkzeug *Association*, das Werkzeug *Aggregation* gewählt.	*Maus: 10* *Tastatur: 3* *13*
21 Anlegen der Assoziation zwischen Bestellsystem und Lagerverwaltungssystem	Analog Punkt 17	*Maus: 9* *Tastatur: 3* *12*
22 Anlegen der Assoziation zwischen Bestellung und Bestellsystem	Analog Punkt 17	*Maus: 10* *Tastatur: 3* *13*
23 Anlegen der Notiz samt Verbindung mit der Klasse Bestellung	Auswählen des *Note*-Werkzeugs aus der Werkzeugleiste, anlegen des Textfeldes im Diagramm und Eingabe des Textes. Vergrößern des Textfeldes per Drag&Drop, damit der Text vollständig sichtbar wird. Auswählen des *Note Connector*-Werkzeugs und verbinden der Klasse Bestellung mit der Notiz.	*Maus: 5* *Tastatur: 1* *6*
24 Anlegen der beiden anderen Notizen	Analog Punkt 23	*Maus: 10* *Tastatur: 2* *12*
25 Anlegen der Abhängigkeit zwischen den beiden Paketen	Auswählen des *Dependency or Instantiates*-Werkzeugs aus der Werkzeugleiste, LK auf Bestellwesen und ziehen nach Lagerverwaltung. LK auf das Symbol *Text Box* in der Werkzeugleiste, plazieren des Textes auf der Abhängigkeit und Eingabe der Zeichenkette <<*access*>>. Dies muß händisch geschehen, da Rose 98 keine Stereotypen für Abhängigkeiten zuläßt.	*Maus: 4* *Tastatur: 1* *5*
Gesamt		***224***

6.3.8 Klassendiagramm - SELECT Enterprise

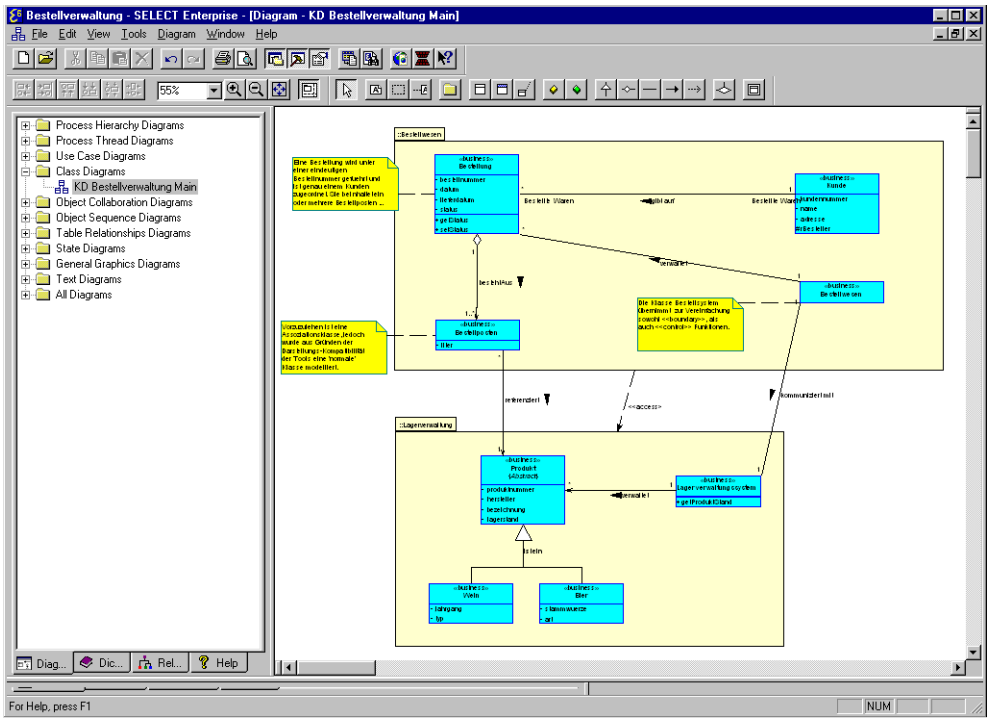

Abb. 6–7
Klassendiagramm in SELECT Enterprise

Schritt	Vorgangsweise	Benötigte Aktionen
01 Anlegen eines neuen Klassendiagramms	RK im Diagram-Explorer auf den Eintrag Class Diagrams, danach Auswahl des Befehls *New Class Diagram* im Kontextmenü. LK im Explorer auf den Namen des neu angelegten Diagramms. DK im Property Window auf den vorgegebenen Bezeichner des neuen Klassendiagramms, Eingabe des gewünschten Namens.	*Maus: 3* *Tastatur: 1* *4*
02 Anlegen der Pakete	LK in der Werkzeugleiste auf das Package-Symbol. Durch Halten der STRG-Taste können mehrere Packages nacheinander durch LK im Diagram-Window angelegt werden. Um diese Funktion wieder auszuschalten, muß anschließend das Pfeilsymbol in der Werkzeugleiste angewählt werden.	*Maus: 4* *Tastatur: 0* *4*
03 Vergeben von Paket-Bezeichnern	LK auf das entsprechende Package. Im Property Window wird wiederum der gewünschte Bezeichner eingetragen.	*Maus: 4* *Tastatur: 2* *6*

Schritt	Vorgangsweise	Benötigte Aktionen
04 Anlegen der Klassen	LK auf das Class-Symbol in der Werkzeugleiste. Mittels gedrückter STRG-Taste werden sämtliche benötigten Klassen eingezeichnet. Um die Klassen einem bestimmten Paket zuzuordnen, wird die entsprechende Klasse direkt in das Paket-Symbol abgelegt. Hierzu muß dieses erst vergrößert werden. Durch Anwählen des Pfeilsymbols wird das wiederholte Einfügen neuer Klassen abgeschlossen.	*Maus:* 12 *Tastatur:* 0 12
05 Vergeben von Klassenbezeichnern, sowie Stereotypen	Vergeben der Bezeichner analog Punkt 03. Zusätzlich noch Vergabe von Stereotypen im Property Window (Eintrag von *boundary* und *control* als Beschreibung - Reiter Custom, *abstract* kann unter dem Reiter *Options* ausgewählt werden.)	*Maus:* 20 *Tastatur:* 10 30
06 Anlegen der Attribute der Klasse Bestellung	Analog Punkt 02. LK auf Symbol *Attribute* in der Werkzeugleiste. STRG-Taste gedrückt halten und mehrere Attribute in die Klasse durch LK anlegen. Vergeben des Bezeichners durch LK auf Attribut, DK im Property Dialog auf Bezeichner und anschließender Eintrag. Datentypen und Initialwerte werden im Ordner *Data Type* eingetragen.	*Maus:* 26 *Tastatur:* 12 38
07 Anlegen der Attribute der Klasse Kunde	Analog Punkt 06	*Maus:* 20 *Tastatur:* 9 29
08 Anlegen der Attribute der Klasse Bestellposten	Analog Punkt 06, nur kann bei Anlegen eines Attributs (ohne STRG-Taste) der Bezeichner direkt eingegeben werden, Datentyp und Initialwert wieder über Property Window.	*Maus:* 5 *Tastatur:* 3 8
09 Anlegen der Attribute der Klasse Produkt	Analog Punkt 06	*Maus:* 26 *Tastatur:* 12 38
10 Anlegen der Attribute der Klasse Wein	Analog Punkt 06	*Maus:* 14 *Tastatur:* 6 20
11 Anlegen der Attribute der Klasse Bier	Analog Punkt 06	*Maus:* 13 *Tastatur:* 5 18

Schritt	Vorgangsweise	Benötigte Aktionen
12 Anlegen der Operationen der Klasse Bestellung	LK auf Symbol *Operation* in der Werkzeugleiste. LK in Klasse, Bezeichner kann sofort vergeben werden. Signatur (*Associated Items*) und Rückgabewert (*Return*) werden wieder im Property Window eingetragen.	*Maus:* 10 *Tastatur:* 6 16
13 Anlegen der Operationen der Klasse Lagerverwaltungssystem	Analog Punkt 13	*Maus:* 6 *Tastatur:* 3 9
14 Anlegen der Assoziation *gibt auf* zwischen Kunde und Bestellung	LK auf das Symbol *Association* in der Werkzeugleiste. LK in die Klasse *Kunde*, LK in die Klasse Bestellung - Assoziation wird angelegt. Vergabe des Bezeichners kann sofort erfolgen, Rollennamen, sowie Kardinalität werden im Property Window festgelegt (*Start Role* und *End Role*). Die Kardinalität wird defaultmäßig mit *1 zu ** festgelegt, wodurch in diesem Fall keine Änderungen notwendig sind.	*Maus:* 8 *Tastatur:* 3 11
15 Anlegen der Assoziation *referenziert* zwischen Bestellposten und Produkt	Analog Punkt 14, die Kardinalität muß geändert werden, Rollennamen werden keine vergeben. Weiters ist die Navigierbarkeit unter der Karteikarte *Options* zu ändern.	*Maus:* 11 *Tastatur:* 1 12
16 Anlegen der Assoziation *verwaltet* zwischen Lagerverwaltungssystem und Produkt	Analog Punkt 14, jedoch muß nur der Bezeichner und die Navigierbarkeit eingetragen werden.	*Maus:* 5 *Tastatur:* 1 6
17 Anlegen der Assoziation *kommuniziert mit* zwischen Lagerverwaltungssystem und Bestellsystem	Analog Punkt 14, die Kardinalität von Lagerverwaltungssystem ist zu ändern.	*Maus:* 6 *Tastatur:* 1 7
18 Anlegen der Assoziation *verwaltet* zwischen Bestellsystem und Bestellung	Analog Punkt 14, jedoch muß nur der Bezeichner eingetragen werden.	*Maus:* 3 *Tastatur:* 1 4

Schritt	Vorgangsweise	Benötigte Aktionen
19 Anlegen der Aggregation *bestehtAus* zwischen Bestellung und Bestellposten	LK auf das Symbol *Aggregation* in der Werkzeugleiste. LK in die Klasse Bestellung, LK in die Klasse Bestellposten - Aggregation wird angelegt, der Bezeichner kann sofort vergeben werden. Die Kardinalität von Bestellposten und die Navigierbarkeit müssen anschließend im Property Window geändert werden.	*Maus: 8* *Tastatur: 1* *9*
20 Anlegen der Generalisation zwischen Produkt und Wein	LK auf das Symbol *Generalization* in der Werkzeugleiste. LK in die Klasse Produkt, LK in die Klasse Wein - Generalisation wird angelegt. Das Tool verlangt auch die Eingabe eines Bezeichners (Eingabe *ist ein*).	*Maus: 3* *Tastatur: 1* *4*
21 Anlegen der Generalisation zwischen Produkt und Bier	Analog Punkt 19. Nach LK in die Klasse Bier fragt das Tool in einem Dialogfenster, ob die vorhandene Generalisation *ist ein* verwendet, oder ob eine neue (*New*) angelegt werden soll. Um einen *Shared Target Style* der beiden Generalisationen zu erreichen, genügt ein RK auf eine Generalisation, danach LK im Kontextmenü auf *Align Generalizations*.	*Maus: 7* *Tastatur: 0* *7*
22 Anlegen der Abhängigkeits-beziehung zwischen den Paketen	LK auf das Symbol *Dependency* in der Werkzeugleiste. LK in das Paket Bestellwesen, LK in das Paket Lagerverwaltung - Beziehung wird angelegt. Sofortige Angabe des Bezeichners (<<*access*>>, obwohl dies eigentlich ein Stereotyp wäre, siehe Punkt 05).	*Maus: 3* *Tastatur: 1* *4*
23 Anlegen der Notizen	LK auf Symbol *Note* in der Werkzeugleiste. STRG-Taste gedrückt halten und mehrere Notizen durch LK im Diagramm anlegen. Anschließend LK auf das Pfeilsymbol in der Werkzeugleiste, um den Vorgang zu beenden.	*Maus: 5* *Tastatur: 0* *5*
24 Eintragen der Notizeninhalte	LK auf die Notiz, DK im Property-Window in das Textfeld, Eingabe des Inhalts	*Maus: 6* *Tastatur: 3* *9*
25 Verbinden der Notiz mit den entsprechenden Klassen	LK auf das Symbol *Link Note* in der Werkzeugleiste. STRG halten, LK in die Notiz, LK in die Klasse - Verbindung wird angelegt. Beenden durch LK auf das Pfeilsymbol in der Werkzeugleiste.	*Maus: 8* *Tastatur: 0* *8*
Gesamt		***318***

6.3.9 Klassendiagramm - StP/UML

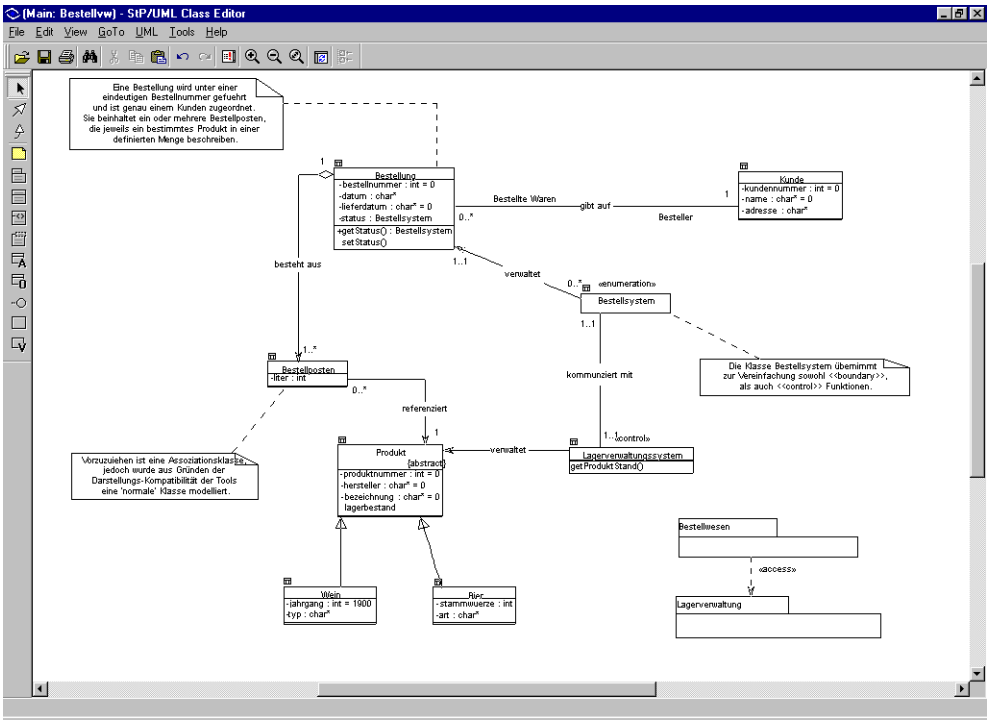

Abb. 6–8
Klassendiagramm in StP/UML

Schritt	Vorgangsweise	*Benötigte Aktionen*
01 Anlegen eines neuen Klassendiagramms	LK im Desktop-Explorer auf den Eintrag *Class*, danach RK und Auswahl des Befehls *New* im Kontextmenü. Der Diagrammeditor wird gestartet. Auswahl des Punkts *File - Save* in der Menüzeile, um das neue Klassendiagramm unter einem Bezeichner abzuspeichern.	*Maus: 6* *Tastatur: 1* *7*
02 Anlegen der Pakete mit Bezeichnern	DK in der Werkzeugleiste auf das Package-Symbol. Durch DK (anstatt LK) können mehrere Packages nacheinander durch LK im Diagrammeditor angelegt werden. Um diese Funktion wieder auszuschalten, muß anschließend das Pfeilsymbol in der Werkzeugleiste angewählt werden. Die Bezeichner können sofort nach Anlegen des entsprechenden Symbols eingetragen werden.	*Maus: 4* *Tastatur: 2* *6*
03 Anlegen der Klassen mit Bezeichnern	Analog Punkt 02, Auswahl des Symbols *Class* in der Werkzeugleiste und Anlegen aller Klasse mit Bezeichnern. Um die Klassen einem bestimmten Paket zuzuordnen, wird die entsprechende Klasse direkt in das Paket-Symbol abgelegt. Hierzu muß dieses erst vergrößert werden.	*Maus: 12* *Tastatur: 8* *20*

Schritt	Vorgangsweise	Benötigte Aktionen
04 Vergeben des Stereotyp *abstract* für die Klasse Produkt	RK auf das Klassensymbol, LK im Kontextmenü auf den Eintrag *Properties*. Auswahl des Typs *abstract* im Dialogfenster und Bestätigung.	*Maus: 4* *Tastatur: 0* *4*
05 Vergeben der restlichen Stereotypen	Analog Punkt 04, jedoch müssen die Stereotypen *control* und *boundary* im Textfeld eingetragen werden.	*Maus: 8* *Tastatur: 2* *10*
06 Anlegen aller Attribute mit Bezeichnern	DK in der Werkzeugleiste auf das *Attribute*-Symbol. Anlegen der Attribute durch LK in die entsprechende Klasse und Vergeben des Bezeichners. Beenden durch LK auf das Pfeilsymbol in der Werkzeugleiste.	*Maus: 18* *Tastatur: 16* *34*
07 Eintragen der Attribut-Eigenschaften der Klasse Bestellung	Um die weiteren Eigenschaften, wie Typ und Initialwert eintragen zu können, muß der sog. *Class Table Editor* geöffnet werden, der die Inhalte (Attribute und Operationen) einer Klasse in Form einer Tabelle darstellt (LK auf die Klasse, dann Auswahl in der Menüleiste Punkt *GoTo*, Unterpunkt *Class Table*). In den einzelnen Textfeldern können nun die erforderlichen Angaben eingetragen werden. Ist noch kein solcher Class-Table vorhanden, wird vorher noch abgefragt, ob ein neuer angelegt werden soll. Abschließend wird der Table gespeichert und das Editor-Fenster verlassen.	*Maus: 13* *Tastatur: 7* *20*
08 Eintragen der Attribut-Eigenschaften der Klasse Kunde	Analog Punkt 07	*Maus: 12* *Tastatur: 6* *18*
09 Eintragen der Attribut-Eigenschaften der Klasse Bestellposten	Analog Punkt 07	*Maus: 8* *Tastatur: 2* *10*
10 Eintragen der Attribut-Eigenschaften der Klasse Produkt	Analog Punkt 07	*Maus: 14* *Tastatur: 8* *22*

Schritt	Vorgangsweise	*Benötigte Aktionen*
11 Eintragen der Attribut-Eigenschaften der Klasse Wein	Analog Punkt 07	*Maus: 10* *Tastatur: 4* *14*
12 Eintragen der Attribut-Eigenschaften der Klasse Bier	Analog Punkt 07	*Maus: 9* *Tastatur: 3* *14*
13 Anlegen aller Operationen mit Bezeichnern	DK in der Werkzeugleiste auf das *Operation*-Symbol. Anlegen der Operationen durch LK in die entsprechende Klasse und Vergeben des Bezeichners. Beenden durch LK auf das Pfeilsymbol in der Werkzeugleiste.	*Maus: 5* *Tastatur: 3* *8*
14 Eintragen der Operations-eigenschaften der Klasse Bestellung	Analog Punkt 07 werden nun im *Class Table Editor* die Operationseigenschaften eingetragen (Signatur, Rückgabewert).	*Maus: 8* *Tastatur: 3* *11*
15 Eintragen der Operations-eigenschaften der Klasse Lagerverwaltungssystem	Analog Punkt 14	*Maus: 8* *Tastatur: 2* *10*
16 Anlegen der Assoziation *gibt auf* zwischen Kunde und Bestellung	Da nicht direkt ein bestimmter Beziehungstyp (Assoziation, Aggregation, Generalisierung, etc.) ausgewählt werden kann, muß im Menü *Tools - Options* - Karteikarte *Default Arc* erst der zu verwendende Typ bestimmt werden. Der Typ kann jedoch für eine Aggregation auch im Nachhinein geändert werden (nach Anlegen einer Beziehung), sodaß die Vorgangsweise in diesem Fall effizienter scheint, daß zuerst alle Beziehungen als Assoziationen modelliert werden und dann der Typ für die Aggregation geändert wird. Anlegen der Beziehung: DK auf Symbol *Arc* in der Werkzeugleiste, LK in die Quell-Klasse, LK in die Ziel-Klasse.	*Maus: 9* *Tastatur: 0* *9*

Schritt	Vorgangsweise	Benötigte Aktionen
17 Anlegen der Assoziation *referenziert* zwischen Bestellposten und Produkt	Nur Anlegen der Beziehung (siehe Punkt 16)	*Maus: 2* *Tastatur: 0* *2*
18 Anlegen der Beziehung *bestehtAus* zwischen Bestellung und Bestellposten	Analog Punkt 17	*Maus: 2* *Tastatur: 0* *2*
19 Anlegen der Assoziation *verwaltet* zwischen Lagerverwaltungssystem und Produkt	Analog Punkt 17	*Maus: 2* *Tastatur: 0* *2*
20 Anlegen der Assoziation *kommuniziert mit* zwischen Lagerverwaltungssystem und Bestellsystem	Analog Punkt 17	*Maus: 2* *Tastatur: 0* *2*
21 Anlegen der Assoziation *verwaltet* zwischen Bestellsystem und Bestellung	Analog Punkt 17	*Maus: 2* *Tastatur: 0* *2*
22 Anlegen der Beziehung zwischen den Paketen	Analog Punkt 17, Abschließen durch LK auf Pfeilsymbol in der Werkzeugleiste.	*Maus: 3* *Tastatur: 0* *3*
23 Anlegen der Generalisation zwischen Produkt und Wein	Hierzu muß im Menü *Tools* - *Options* - Karteikarte *Default Arc* erst wieder der zu verwendende Typ (*Generalization*) ausgewählt werden. Einzeichnen der Beziehung erfolgt analog zu den anderen Typen.	*Maus: 9* *Tastatur: 0* *9*

Schritt	Vorgangsweise	Benötigte Aktionen
24 Anlegen der Generalisation zwischen Produkt und Bier	Analog Punkt 22, Abschließen durch LK auf Pfeilsymbol in der Werkzeugleiste.	*Maus:* 3 *Tastatur:* 0 3
25 Bestimmen der Eigenschaften der Assoziation *gibt auf* zwischen Kunde und Bestellung	RK auf Assoziation, LK auf Punkt *Properties* im Kontextmenü. Eintragen des Bezeichners und der Multiplizitäten in Textfelder im Dialogfenster und Bestätigung.	*Maus:* 5 *Tastatur:* 3 8
26 Anlegen von Rollennamen für die Assoziation *gibt auf* zwischen Kunde und Bestellung	LK auf Assoziation, Auswahl des Menüs *UML - Create Association Roles*. Es erscheinen zwei Textfelder an beiden Enden der Assoziation, in die der gewünschte Rollenname eingetragen wird.	*Maus:* 6 *Tastatur:* 2 8
27 Bestimmen der Eigenschaften der Aggregation *bestehtAus* zwischen Bestellung und Bestellposten	Analog Punkt 25, im Dialogfenster muß auch der Typ *Aggregation Type* festgelegt werden. Weiters muß die Navigierbarkeit eingetragen (ausgewählt) werden.	*Maus:* 8 *Tastatur:* 3 11
28 Bestimmen der Eigenschaften der Assoziation *referenziert* zwischen Bestellposten und Produkt	Analog Punkt 25, ebenfalls Auswahl der entsprechenden Navigierbarkeit.	*Maus:* 6 *Tastatur:* 3 9
29 Bestimmen der Eigenschaften der Assoziation *verwaltet* zwischen Lagerverwaltungssystem und Produkt	Analog Punkt 25	*Maus:* 5 *Tastatur:* 3 8

Schritt	Vorgangsweise	Benötigte Aktionen
30 Bestimmen der Eigenschaften der Assoziation *kommuniziert* mit zwischen Lagerverwaltungssystem und Bestellsystem	Analog Punkt 25	*Maus:* 5 *Tastatur:* 3 *8*
31 Bestimmen der Eigenschaften der Assoziation *verwaltet* zwischen Bestellsystem und Bestellung	Analog Punkt 25, ebenfalls Auswahl der entsprechenden Navigierbarkeit	*Maus:* 6 *Tastatur:* 3 *9*
32 Bestimmen der Eigenschaften der Abhängigkeitsbeziehung zwischen den Paketen	Analog Punkt 25, es wird lediglich der Stereotyp *access* eingetragen.	*Maus:* 4 *Tastatur:* 1 *5*
33 Anlegen der Notizen	DK auf das Symbol *Note* in der Werkzeugleiste. LK in das Klassendiagramm an der gewünschten Stelle. Beenden durch LK auf das Pfeilsymbol in der Werkzeugleiste.	*Maus:* 5 *Tastatur:* 0 *5*
34 Eintragen der Notizeninhalte	DK auf die entsprechende Notiz und Eingabe des Inhalts.	*Maus:* 3 *Tastatur:* 3 *6*
35 Verbinden der Notizen mit den entsprechenden Klassen	DK auf das Symbol *Arc* in der Werkzeugleiste. LK in die Notiz, LK in die Klasse - Verbindung wird angelegt. Beenden durch LK auf das Pfeilsymbol in der Werkzeugleiste.	*Maus:* 5 *Tastatur:* 0 *5*
Gesamt		***322***

6.3.10 Klassendiagramm - WithClass

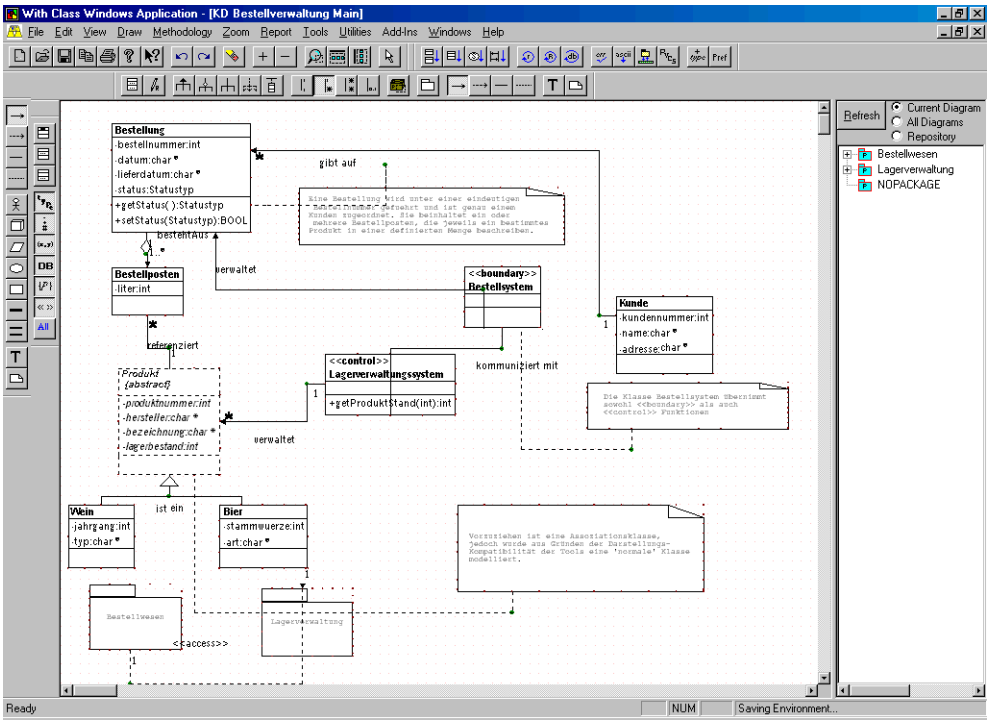

Abb. 6–9
Klassendiagramm in WithClass

Schritt	Vorgangsweise	*Benötigte Aktionen*
01 Anlegen eines neuen Klassendiagramms	RK im Project-Explorer auf den Eintrag *Class*, danach Auswahl des Befehls *New* im Kontextmenü. Eingabe des gewünschten Diagrammbezeichners.	*Maus: 2* *Tastatur: 1* *3*
02 Anlegen der Pakete	LK in der Werkzeugleiste auf das Symbol *Package Shape*. LK im Diagrammfenster, um das Paket anzulegen. Ein Dialog wird automatisch geöffnet, in welchem der Bezeichner eingetragen und bestätigt wird.	*Maus: 6* *Tastatur: 2* *8*
03 Anlegen der Klassen	LK in der Werkzeugleiste auf das Symbol *Class*. LK im Diagrammfenster, um die neue Klasse anzulegen. Der Bezeichner kann direkt eingetragen werden.	*Maus: 16* *Tastatur: 8* *24*
04 Zuordnen der Klassen zu den Paketen	DK im Repository-Browser auf das gewünschte Paket. Im Dialogfenster LK auf Reiter Classes, dann LK auf Button *Add*. Jede Klasse muß im nächsten Dialog einzeln ausgewählt (keine Mehrfachselektion) und das Hinzufügen durch LK auf *OK* bestätigt werden.	*Maus: 22* *Tastatur: 0* *22*

Schritt	Vorgangsweise	Benötigte Aktionen
05 Vergeben der Stereotypen der Klassen	DK auf die entsprechende Klasse, Auswahl der Karteikarte *Specs 2* im Dialogfenster. Eintrag der Stereotypen *control* und *boundary* erfolgt im entsprechenden Textfeld, für *abstract* steht ein Auswahlfeld zur Verfügung.	*Maus: 12* *Tastatur: 2* *14*
06 Anlegen der Attribute der Klasse Bestellung	DK auf die Klasse, Auswahl der Karteikarte *Attributes* im Dialogfenster. Der Bezeichner des neuen Attributs kann in einem Textfeld eingegeben und bestätigt werden. So werden zunächst alle Attribute mit ihren Bezeichnern angelegt. Danach müssen noch die Details (Typ und Initialwert) nachgetragen werden: DK auf Bezeichner, Eintrag der Werte im folgenden Dialogfenster. Als Defaultwerte sind der Typ *int* und der Initialwert *0* bereits voreingestellt, sodaß bei manchen Attributen diese Aktionen entfallen können. Auch Standardtypen, wie *char** können direkt ausgewählt werden.	*Maus: 13* *Tastatur: 5* *18*
07 Anlegen der Attribute der Klasse Kunde	Analog Punkt 06	*Maus: 11* *Tastatur: 3* *14*
08 Anlegen der Attribute der Klasse Bestellposten	Analog Punkt 06	*Maus: 3* *Tastatur: 1* *4*
09 Anlegen der Attribute der Klasse Produkt	Analog Punkt 06	*Maus: 11* *Tastatur: 4* *15*
10 Anlegen der Attribute der Klasse Wein	Analog Punkt 06, Initialwert von *jahrgang* muß jedoch geändert werden.	*Maus: 10* *Tastatur: 3* *13*
11 Anlegen der Attribute der Klasse Bier	Analog Punkt 06	*Maus: 7* *Tastatur: 2* *9*

Schritt	Vorgangsweise	Benötigte Aktionen
12 Anlegen der Operationen der Klasse Bestellung	Das Anlegen der Operationen erfolgt analog zum Anlegen von Attributen: DK auf die Klasse, Auswahl der Karteikarte *Operations* im Dialogfenster. Es werden zunächst alle Operationen mit ihren Bezeichnern angelegt. Es fehlen noch Rückgabetypen und Parameter: DK auf den Bezeichner, Eintrag der Werte im folgenden Dialogfenster (Rückgabewert unter *Operation Specification* und Parameterliste unter *Parameters*).	*Maus:* 15 *Tastatur:* 3 18
13 Anlegen der Operationen der Klasse Lagerverwaltungssystem	Analog Punkt 12	*Maus:* 11 *Tastatur:* 2 13
14 Anlegen der Assoziation *gibt auf* zwischen Kunde und Bestellung	LK auf das Symbol *Association* in der Werkzeugleiste. Außerdem LK auf das Symbol in der Werkzeugleiste, welches die gewünschte Kardinalität repräsentiert. LK in die Klasse Kunde, LK in die Klasse Bestellung - Assoziation wird angelegt. Vergabe des Bezeichners kann sofort erfolgen, Rollennamen können nicht vergeben werden, im Eigenschaftendialog der Assoziation (DK) wird die Bezeichnung *Role Name* für den Assoziationsbezeichner verwendet, also ist hier keine weitere Aktion erforderlich.	*Maus:* 4 *Tastatur:* 1 5
15 Anlegen der Assoziation *referenziert* zwischen Bestellposten und Produkt	Analog Punkt 14, Kardinalität ist bereits richtig ausgewählt.	*Maus:* 3 *Tastatur:* 1 4
16 Anlegen der Assoziation *verwaltet* zwischen Lagerverwaltungssystem und Produkt	Analog Punkt 14	*Maus:* 3 *Tastatur:* 1 4
17 Anlegen der Assoziation *kommuniziert mit* zwischen Lagerverwaltungssystem und Bestellsystem	Analog Punkt 14	*Maus:* 3 *Tastatur:* 1 4

Schritt	Vorgangsweise	Benötigte Aktionen
18 Anlegen der Assoziation *verwaltet* zwischen Bestellsystem und Bestellung	Analog Punkt 14	*Maus:* 3 *Tastatur:* 1 4
19 Anlegen der Aggregation *bestehtAus* zwischen Bestellung und Bestellposten	LK auf das Symbol *Aggregation* in der Werkzeugleiste. LK in die Klasse Bestellung, LK in die Klasse Bestellposten - Aggregation wird angelegt. Vergabe des Bezeichners, danach noch Änderung der Kardinalität von * auf *1..** im Eigenschaftendialog (DK) der Aggregation.	*Maus:* 6 *Tastatur:* 2 8
20 Anlegen der Generalisation zwischen Produkt und Wein	LK auf das Symbol *Generalization* in der Werkzeugleiste. LK in die Klasse Produkt, LK in die Klasse Wein - Generalisation wird angelegt. Das Tool verlangt auch die Eingabe eines Bezeichners (Eingabe *ist ein*).	*Maus:* 3 *Tastatur:* 1 4
21 Anlegen der Generalisation zwischen Produkt und Bier	Analog Punkt 20. Um eine Art *Shared Target Style* der beiden Generalisationen zu erreichen, erfolgt nach Auswahl des Symbols ein LK in einen Punkt unterhalb der Pfeilspitze der vorigen Generalisation, danach LK in Klasse Bier.	*Maus:* 3 *Tastatur:* 0 3
22 Anlegen der Abhängigkeits-beziehung zwischen den Paketen	LK auf das Symbol *Dependency* in der Werkzeugleiste. LK in das Paket Bestellwesen, LK in das Paket Lagerverwaltung - Beziehung wird angelegt. DK auf die Dependency, um im Eigenschaftendialog den Stereotyp *access* einzugeben.	*Maus:* 7 *Tastatur:* 1 8
23 Anlegen der Notizen und Eintragen der Inhalte	LK auf das Symbol *Note Shape* in der Werkzeugleiste. LK in das Klassendiagramm an der gewünschten Stelle. Eingabe des Inhalts.	*Maus:* 6 *Tastatur:* 3 9
24 Verbinden der Notizen mit den entsprechenden Klassen	LK auf das Symbol *Dashed Line* in der Werkzeugleiste. LK in die Notiz, LK in die Klasse - Verbindung wird angelegt.	*Maus:* 9 *Tastatur:* 0 9
Gesamt		***236***

6.4 Erstellung des Use-Case- und Sequenzdiagramms

6.4.1 Zusammenfassung der Ergebnisse

In Abb. 6–10 können die Ergebnisse der Analyse der Erstellung von Use-Case- bzw. Sequenzdiagramm abgelesen werden.

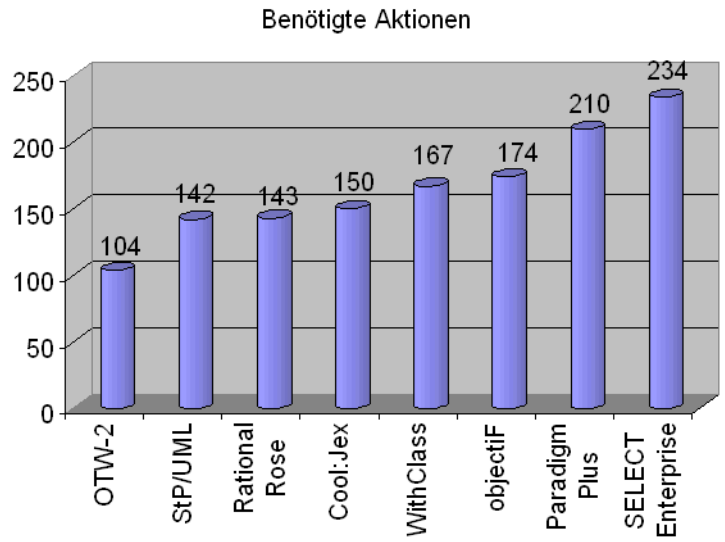

Abb. 6–10
Ergebnisse der UCD/ SD-Erstellung

6.4.2 Interpretation der Ergebnisse

Bei der Erstellung von Use-Case- und Sequenzdiagramm liegt der Großteil der Programme auf einem Niveau. Lediglich OTW-2, Paradigm Plus und SELECT Enterprise weichen signifikant von diesem ab.

Die gute Leistung von OTW-2 wird durch den Umstand relativiert, daß im Sequenzdiagramm kein Aktor angelegt werden konnte und daher auch alle Nachrichten wegfielen, die mit diesem in Verbindung stehen sollten.

In Paradigm Plus müssen die Lebenslinien im Sequenzdiagramm händisch verlängert werden. Dies ist während der Konstruktion des Diagramms mehrmals für alle Objekte nötig und bedeutet einen nicht unwesentlichen Mehraufwand.

Bei SELECT Enterprise muß für jede Nachricht erst eine Sequenz angelegt werden, die eine kurze Beschreibung enthalten kann. Obwohl dieses Konzept interessant scheint und auch durchaus seine Vorteile haben mag, verlangt es vom Benutzer einige zusätzliche Bearbeitungsschritte.

6.4.3 Use-Case-/Sequenzdiagramm - Cool:Jex

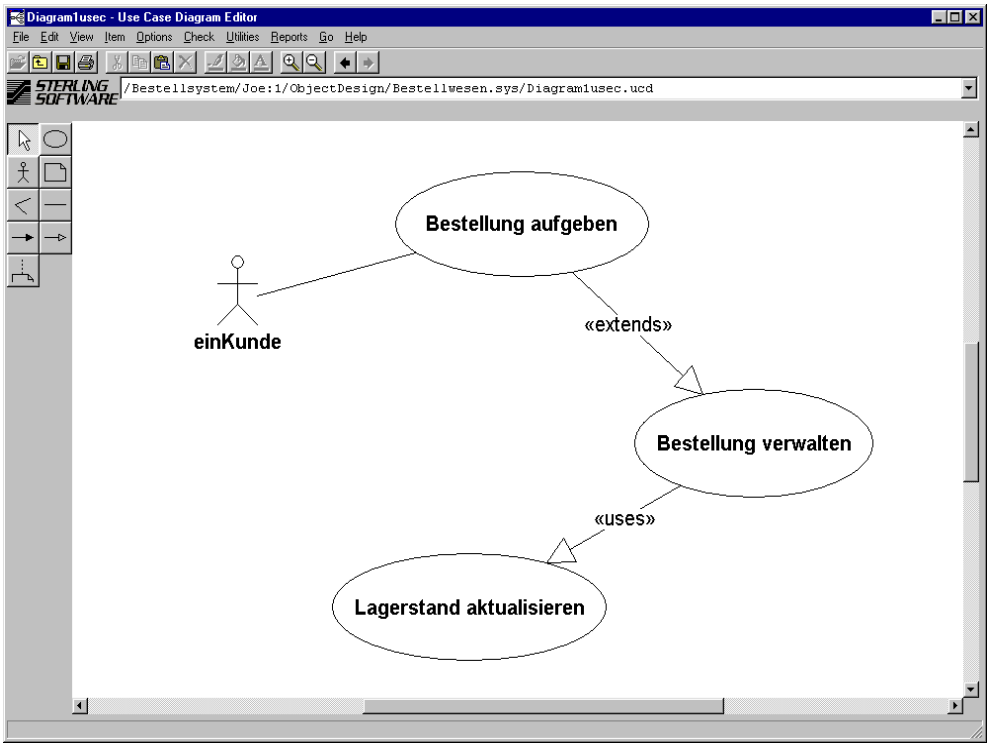

Abb. 6–11
Use-Case Diagramm in Cool:Jex

Schritt	Vorgangsweise	*Benötigte Aktionen*
01 Anlegen eines UseCase Diagramms	RK auf ein Paket im Browser und Auswahl von *New Diagram* aus dem Kontextmenü. Festlegen des Diagrammtyps *Use-Case Diagram* im folgenden Dialog, Setzen des Eingabefokus auf das Namensfeld und Eingabe eines Bezeichners. Nach der Bestätigung mit Enter öffnet sich das neue Diagramm automatisch.	*Maus: 4* *Tastatur: 1* *5*
02 Anlegen des Aktors Kunde	Auswahl des Werkzeugs *Actor*, LK in den Zeichenbereich und Eingabe eines Namens für den neuen Aktor.	*Maus: 2* *Tastatur: 1* *3*
03 Anlegen der drei Use Cases	Auswahl des *Use Case* Werkzeugs aus der Werkzeugleiste. Anlegen des ersten Use Cases mit einem LK auf die Zeichenfläche und Eingabe des Bezeichners. Anlegen der beiden anderen Use Cases auf die gleiche Art. (Hierbei wird der *Sticky Mode* vorausgesetzt, der dafür sorgt, daß das benutzte Werkzeug nicht automatisch deaktiviert wird.)	*Maus: 4* *Tastatur: 3* *7*
04 Anlegen der Assoziationen zwischen den Use Cases	LK auf das *Undirected Communication Association*-Tool in der Werkzeugleiste. Anlegen der Relation zwischen Aktor und *Bestellung aufgeben*. Auswahl des *Use-Case Generalization*-Werkzeugs und Verbinden der drei Use-Cases untereinander.	*Maus: 8* *Tastatur: 0* *8*

Schritt	Vorgangsweise	Benötigte Aktionen
05 Festlegen der Stereotypen der Relationen	RK auf die Beziehungskante und Auswahl von *Edit Properties* aus dem Kontextmenü. Auswählen des gewünschten Stereotyps aus der Droplist *Stereotype* und Schließen des Dialoges mit *ok*. Beide Stereotype werden auf die gleiche Art zugewiesen.	*Maus: 10* *Tastatur: 0* *10*

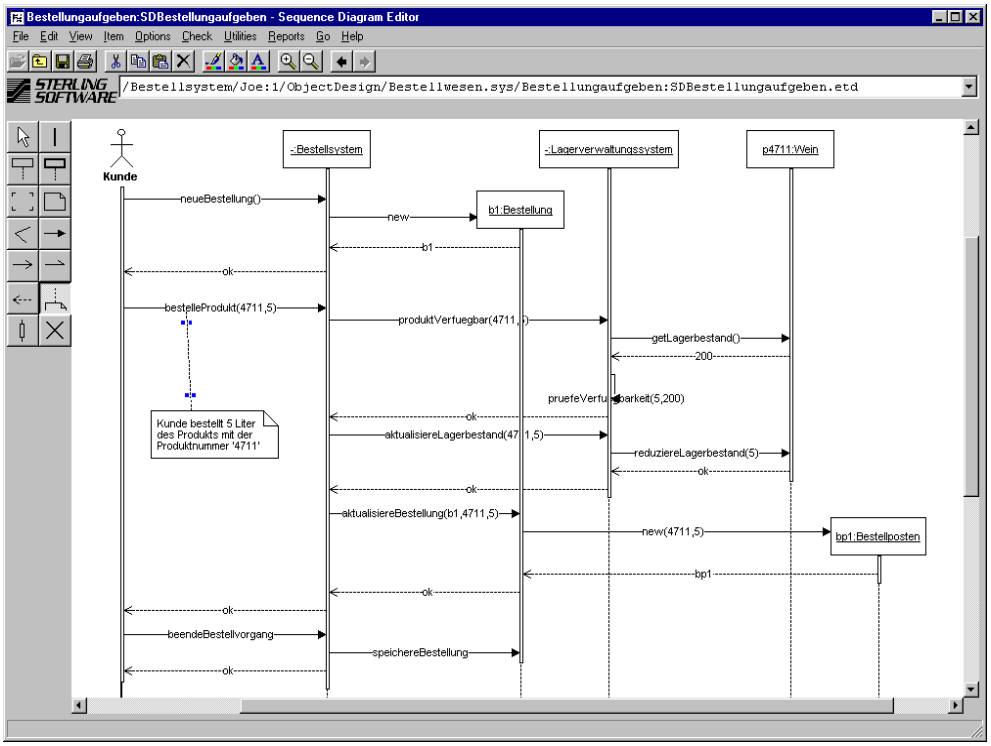

Abb. 6–12
Sequenzdiagramm in Cool:Jex

Schritt	Vorgangsweise	Benötigte Aktionen
06 Anlegen eines Sequenzdiagramms	RK auf den Use-Case *Bestellung aufgeben* im Use-Case Diagramm. Auswählen des Befehls *Open* aus dem Kontextmenü, Anlegen eines Sequenzdiagramms und Eingeben des gewünschten Bezeichners.	*Maus: 3* *Tastatur: 1* *4*
07 Anlegen des Aktors	Auswählen des *Initiator*-Werkzeugs und anlegen des Aktors im Diagramm mit LK. Eingabe des Bezeichners direkt im Diagramm.	*Maus: 2* *Tastatur: 1* *3*
08 Anlegen der 5 Objekte	Auswählen des *Object*-Werkzeugs und Anlegen der Objekte im Diagramm mit LK. Bezeichner und Typ des Objektes werden, durch einen Doppelpunkt getrennt, direkt in das Diagramm eingetragen.	*Maus: 6* *Tastatur: 10* *16*

Schritt	Vorgangsweise	Benötigte Aktionen
09 Anlegen der Messages *neue Bestellung* und *new*, inklusive der Rückgabenachrichten	Auswählen des *Nested Message*-Werkzeugs und Anlegen der Messages, die Funktionsaufrufe darstellen. Der Bezeichner und eine optionale Parameterliste werden direkt im Diagramm eingetragen. Für die Rückgabenachrichten muß das Werkzeug *Return Message* gewählt werden.	*Maus: 10* *Tastatur: 4* *14*
10 Anlegen nächsten 6 Messages (von *bestelle Produkt (4711, 5)* bis zur dazugehörigen Rückgabenachricht)	Analog Punkt 09. Bei der rekursiven Nachricht muß das Werkzeug *Message to self* verwendet werden.	*Maus: 15* *Tastatur: 6* *21*
11 Anlegen der 4 Messages (von *aktualisiereLagerbestand(4711, 5)* bis zur dazugehörigen Rückgabenachricht)	Analog Punkt 09.	*Maus: 10* *Tastatur: 4* *14*
12 Anlegen der verbleibenden 8 Messages	Analog Punkt 09.	*Maus: 18* *Tastatur: 8* *26*
13 Einzeichnen der Lebenslinien der Objekte	Auswählen des *In Scope Region*-Werkzeugs aus der Werkzeugleiste, Einzeichnen der Lebenslinien und "Zurechtziehen" derselben.	*Maus: 13* *Tastatur: 0* *13*
14 Anlegen der Notiz	Auswählen des *Note*-Werkzeugs aus der Werkzeugleiste, Anlegen des Textfeldes im Diagramm und Eingabe des Textes. Auswählen des *Note Connector*-Werkzeugs und Verbinden der Klasse Bestellung mit der Notiz.	*Maus: 5* *Tastatur: 1* *6*
Gesamt		***150***

6.4.4 Use-Case-/Sequenzdiagramm - objectiF

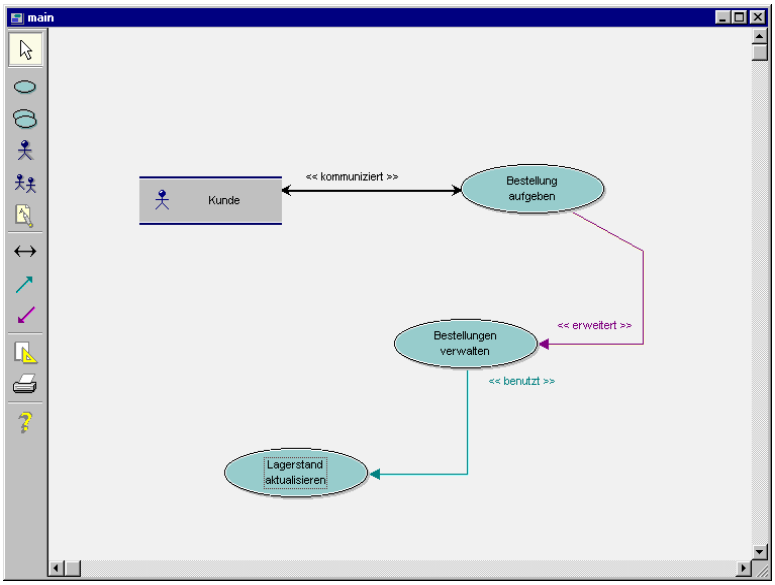

Abb. 6–13
Use-Case Diagramm in objectiF

Schritt	Vorgangsweise	*Benötigte Aktionen*
01 Anlegen eines UseCase Diagramms	Öffnen des Browsers für das Anwendungsfallmodell durch einen Klick auf die entsprechende Schaltfläche im Hauptfenster. RK auf den Knoten *Anwendungsfalldiagramme* und Auswahl von *neues Anwendungsfalldiagramm* aus dem Kontextmenü. Eingabe eines Bezeichners im folgenden Dialog und Bestätigung mit Enter. Öffnen des neuen Diagramms.	*Maus: 5* *Tastatur: 1* *6*
02 Anlegen des Aktors Kunde	Auswahl des Werkzeugs *Akteur anlegen*, LK in den Zeichenbereich und Eingabe eines Namens für den neuen Aktor im folgenden Dialog.	*Maus: 2* *Tastatur: 1* *3*
03 Anlegen der drei Use Cases	Auswahl des *Anwendungsfall anlegen* Werkzeugs aus der Werkzeugleiste. Anlegen des ersten Use Cases mit einem LK auf die Zeichenfläche und Eingabe des Bezeichners. Anlegen der beiden anderen Use Cases auf die gleiche Art.	*Maus: 4* *Tastatur: 3* *7*
04 Anlegen der Relationen zwischen den Use Cases	Für alle drei Beziehungen wird zuerst ein Werkzeug aus der Werkzeugleiste gewählt, und anschließend die beiden entsprechenden Elemente verknüpft. Der Aktor wird mit *Bestellung aufgeben* über eine *Kommuniziert-*, *Bestellung aufgeben* mit *Bestellung verwalten* über eine *Benutzt-* und *Bestellung verwalten* mit *Lagerstand aktualisieren* über eine *Erweitert-*Beziehung verbunden.	*Maus: 9* *Tastatur: 0* *9*

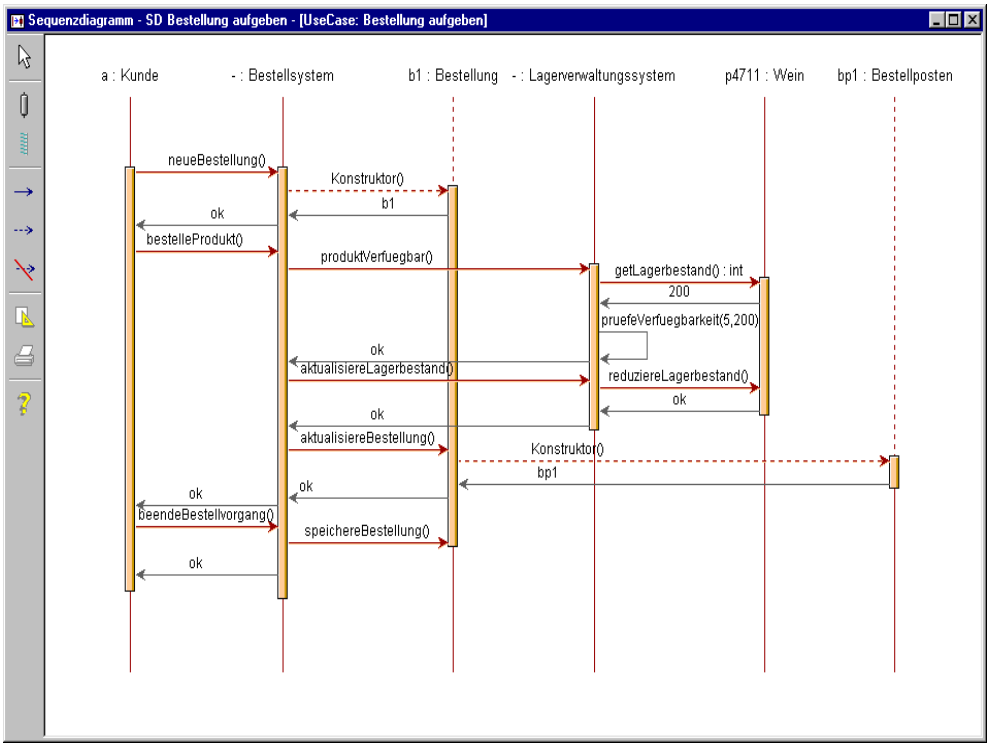

Abb. 6–14
Sequenzdiagramm in objectiF

Schritt	Vorgangsweise	*Benötigte Aktionen*
05 Sequenzdiagramm anlegen	Den Browser für das Anwendungsfallmodell starten und mit einem RK auf den Use-Case *Bestellung aufgeben* ein zugehöriges Sequenzdiagramm erzeugen.	*Maus: 3* *Tastatur: 1* *4*
06 Anlegen der 6 Objekte	Auswählen des *Instanz*-Werkzeugs und anlegen der Objekte im Diagramm mit LK. Im Eigenschaftsdialog, der nach dem Anlegen automatisch gestartet wird, wird eine entsprechende Klasse zugeordnet.	*Maus: 25* *Tastatur: 0* *25*
07 Anlegen der Messages *neue Bestellung* und *new*, inklusive der Rückgabenachrichten	Auswählen des *Message*-Werkzeugs und anlegen der Kanten per Drag&Drop im Diagramm. Im erscheinenden Eigenschaftsdialog wird der Bezeichner der Nachricht eingegeben und der Dialog mit Enter geschlossen. Wird mit der Nachricht eine Methode verknüpft, geschieht dies über die Checkbox *Methode zuordnen*. Hier kann auch über einen Kontextmenübefehl eine neue Methode angelegt werden. Soll ein Konstruktor aufgerufen werden, geschieht dies mit dem *Konstruktor Message*-Werkzeug aus der Werkzeugleiste.	*Maus: 17* *Tastatur: 3* *20*

Schritt	Vorgangsweise	*Benötigte Aktionen*
08 Anlegen nächsten 6 Messages (von *bestelle Produkt (4711, 5)* bis zur dazugehörigen Rückgabenachricht)	Analog Punkt 07.	*Maus: 26* *Tastatur: 8* *34*
09 Anlegen der 4 Messages (von *aktualisiereLagerbestand(4711, 5)* bis zur dazugehörigen Rückgabenachricht)	Analog Punkt 07.	*Maus: 18* *Tastatur: 6* *24*
10 Anlegen der verbleibenden 8 Messages	Analog Punkt 07.	*Maus: 34* *Tastatur: 8* *42*
11 Anlegen der Notiz	**objectiF erlaubt es nicht, Notizen im Sequenzdiagramm anzulegen.**	*Maus: 0* *Tastatur: 0* *0*
Gesamt		***174***

6.4.5 Use-Case-/Sequenzdiagramm - OTW-2

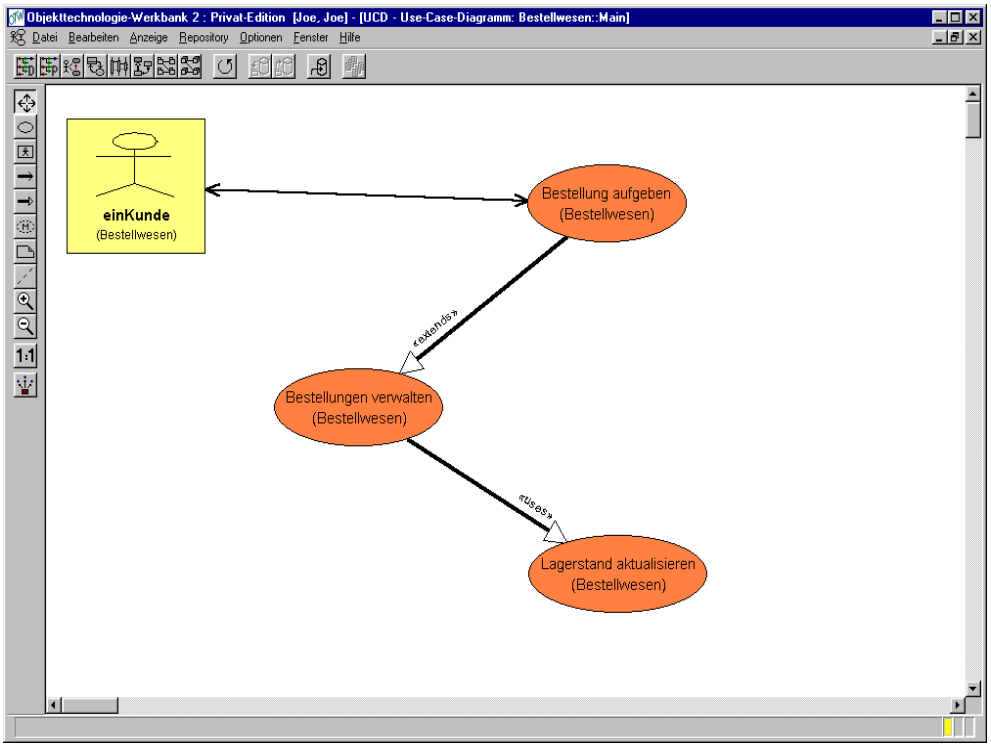

Abb. 6–15
Use-Case Diagramm in OTW-2

Schritt	Vorgangsweise	*Benötigte Aktionen*
01 Anlegen eines UseCase Diagramms	RK auf den Knoten *Use-Case-Diagramm* im Diagramm-Browser. Auswahl des Befehls *neues Diagramm* aus dem Kontextmenü. Auswählen des Paketes, in dem das Diagramm angelegt werden soll im linken Teil des Dialoges *Use-Case Diagramm öffnen*, danach Setzen des Eingabefokus mittels LK in das Textfeld *Use-Case Diagramm*. Eingabe eines Bezeichners und Bestätigen der Eingabe mit Enter. Das neu angelegte Diagramm wird automatisch geöffnet.	*Maus:* 4 *Tastatur:* 1 5
02 Anlegen des Aktors Kunde	Auswahl des Werkzeugs *Akteur*, LK in den Zeichenbereich und Eingabe eines Namens für den neuen Aktor. In diesem Fall ist im linken Teil des Eingabe-Dialoges bereits das richtige Paket gewählt. Bestätigung des Vorganges mit Enter.	*Maus:* 2 *Tastatur:* 1 3
03 Anlegen der drei Use Cases	Auswahl des *Use Case hinzufügen* Werkzeugs aus der Werkzeugleiste. Anlegen des ersten Use Cases mit einem LK auf die Zeichenfläche und Eingabe des Bezeichners im folgenden Dialog. Bestätigung und Schließen des Dialoges mit Enter. Anlegen der beiden anderen Use Cases auf die gleiche Art.	*Maus:* 4 *Tastatur:* 3 7

Schritt	Vorgangsweise	*Benötigte Aktionen*
04 Anlegen der Interaktion zwischen Aktor und *Bestellung aufgeben*	LK auf das *Interaktion hinzufügen*-Tool in der Werkzeugleiste. LK zuerst auf den Aktor, danach auf den Use-Case. Demarkieren der beiden Checkboxen auf der ersten Dialogseite des folgenden Dialoges, Schließen des Fensters mit *ok*.	*Maus:* 6 *Tastatur:* 0 6
05 Anlegen der Beziehung zwischen den Use-Cases	LK auf das Symbol *Use-Case Beziehung* in der Werkzeugleiste. LK zuerst auf den ersten, dann den zweiten Use-Case der jeweiligen Beziehung.	*Maus:* 5 *Tastatur:* 0 5
06 Festlegen der Stereotypen für die beiden Beziehungen	RK auf die Verbindungskante zwischen den Use-Cases und Auswahl des Befehls *Eigenschaften* aus dem Kontextmenü. Wechseln auf die Dialogseite *Stereotype* und Auswählen des gewünschten Stereotyps (extends) in der linken Listbox. Zuweisen des gewählten Stereotyps mit der Schaltfläche >> und Beenden des Dialoges mit *ok*. Festlegen des Stereotyps *uses* für die Beziehung zwischen *Bestellungen verwalten* und *Lagerstand aktualisieren* analog, jedoch muß der Stereotyp *uses* zuerst angelegt werden.	*Maus:* 14 *Tastatur:* 1 15

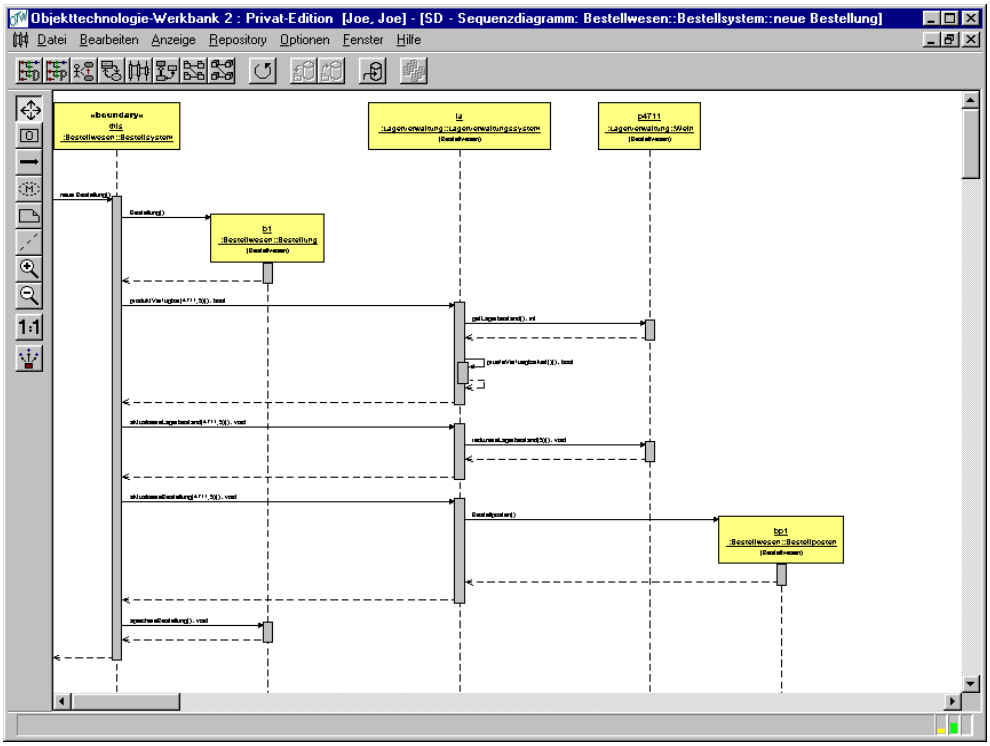
Objekttechnologie-Werkbank 2 : Privat-Edition [Joe, Joe] - [SD - Sequenzdiagramm: Bestellwesen::Bestellsystem::neue Bestellung]
Datei Bearbeiten Anzeige Repository Optionen Fenster Hilfe

Abb. 6–16
Sequenzdiagramm in OTW-2

Schritt	Vorgangsweise	*Benötigte Aktionen*
07 Anlegen eines Sequenzdiagramms	RK auf den Knoten *Sequenzdiagramme* im Browser. Da Sequenzdiagramme in OTW-2 nur Methoden, nicht jedoch Use-Cases zugeordnet werden können, wird für die Klasse Bestellsystem die Funktion *neueBestellung* generiert und dieser das Sequenzdiagramm zugeordnet.	*Maus: 2* *Tastatur: 1* *3*
08 Anlegen des Aktors	**OTW-2 kann keinen Aktor im Sequenzdiagramm darstellen. Dieser und alle mit ihm in Verbindung stehenden Nachrichten werden weggelassen.**	*Maus: 5* *Tastatur: 1* *6*
09 Anlegen des *Bestellung*-Objektes	Auswählen des *Object* Werkzeugs und Anlegen des Objekts im Diagramm mit LK. Festlegen des Bezeichners und des Typs im automatisch erscheinenden Eigenschaftsdialog.	*Maus: 5* *Tastatur: 1* *6*
10 Anlegen der übrigen Objekte im Diagramm	Analog Punkt 08	*Maus: 15* *Tastatur: 3* *18*
11 Anlegen der Message die ein neues Bestellungs-Objekt generiert und der Rückgabenachricht.	Auswählen des *Neue Botschaft*-Werkzeugs und LK zuerst auf die Quelle, dann auf das Ziel der Botschaft. Im nun erscheinenden Eigenschaftsdialog muß eine Methode (in diesem Fall der Konstruktor) angegeben werden. Es können neue Methoden erstellt werden, welche sofort in das Modell übernommen werden. Für jede Botschaft wird automatisch eine Rückgabenachricht generiert.	*Maus: 5* *Tastatur: 0* *5*
12 Anlegen nächsten 5 Messages (von *produktVerfügbar(4711,5)* bis zur dazugehörigen Rückgabenachricht)	Analog Punkt 10. Bei der rekursiven Nachricht muß zweimal auf das selbe Objekt geklickt werden.	*Maus: 19* *Tastatur: 3* *22*
13 Anlegen der 4 Messages (von *aktualisiereLagerbestand(4711, 5)* bis zur dazugehörigen Rückgabenachricht)	Analog Punkt 10.	*Maus: 6* *Tastatur: 2* *8*

Schritt	Vorgangsweise	Benötigte Aktionen
14 Anlegen der verbleibenden 6 Messages	Analog Punkt 10.	*Maus:* 10 *Tastatur:* 2 12
15 Anlegen der Notiz	**Da in OTW-2 der Aktor und alle mit ihm in Verbindung stehenden Messages wegfallen, kann auch die Notiz nicht eingezeichnet werden, die ja an eine dieser Nachrichten geknüpft werden sollte.**	*Maus:* 0 *Tastatur:* 0 0
Gesamt		***104***

6.4.6 Use-Case-/Sequenzdiagramm - Paradigm Plus

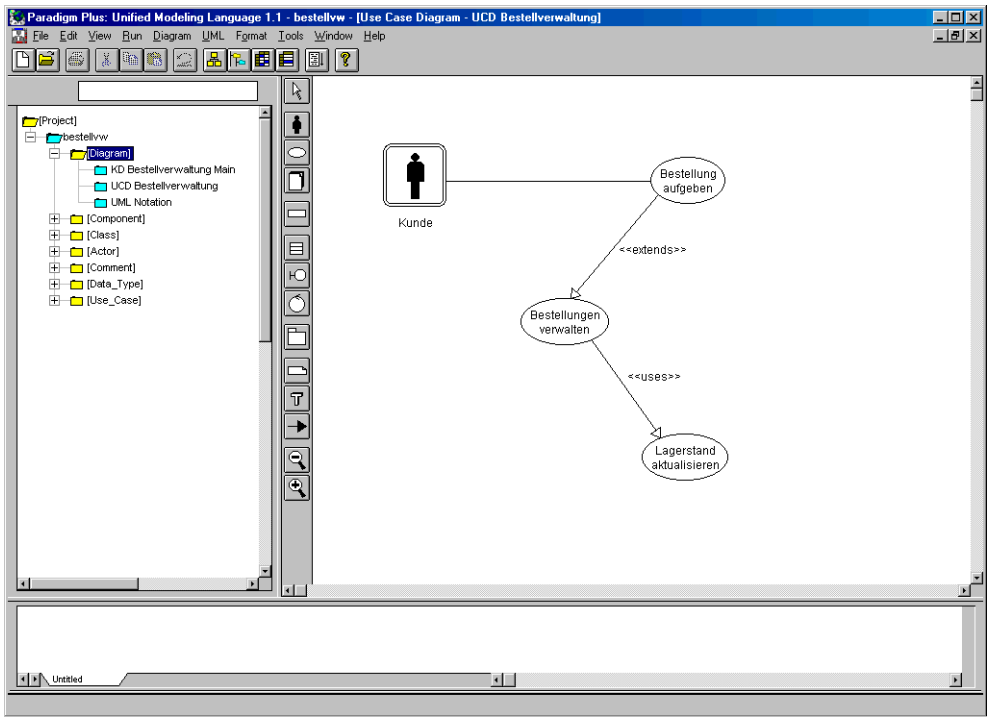

Abb. 6–17
Use-Case Diagramm in Paradigm Plus

Schritt	Vorgangsweise	Benötigte Aktionen
01 Anlegen eines UseCase Diagramms	Auswahl des Menüpunkts *Diagram - New* in der Menüleiste, Eingabe des Bezeichners im Textfeld und LK auf den Typ *Use Case Diagram*. Bestätigung mit *ok*.	*Maus:* 4 *Tastatur:* 1 5

Schritt	Vorgangsweise	*Benötigte Aktionen*
02 Anlegen des Aktors Kunde	Auswahl des Werkzeugs *Actor*, LK in den Zeichenbereich, Aktor wird angelegt. LK auf das Pfeilsymbol, um den Vorgang zu beenden. LK auf den Aktor und Eingabe eines Namens.	*Maus: 4* *Tastatur: 1* *5*
03 Anlegen der drei Use Cases	Auswahl des *Use Case* Werkzeugs aus der Werkzeugleiste. Anlegen aller Use Cases durch LK auf die Zeichenfläche. Beenden durch LK auf Pfeilsymbol. Zweimaliger LK (langsamer als DK) auf jeden Use Case und Eingabe der Bezeichnung.	*Maus: 11* *Tastatur: 3* *14*
04 Anlegen der Assoziationen zwischen den Use Cases	LK auf das *Relationship*-Werkzeug in der Werkzeugleiste. LK in einen Quell-Use Case (in den Aktor), LK in den Ziel-Use Case, Beziehung wird angelegt. Bei der Beziehung zwischen zwei Anwendungsfällen wird noch in einem Dialog der Typ (*extends* / *uses*) ausgewählt. Wieder Beenden durch Auswahl des Pfeilsymbols.	*Maus: 11* *Tastatur: 0* *11*

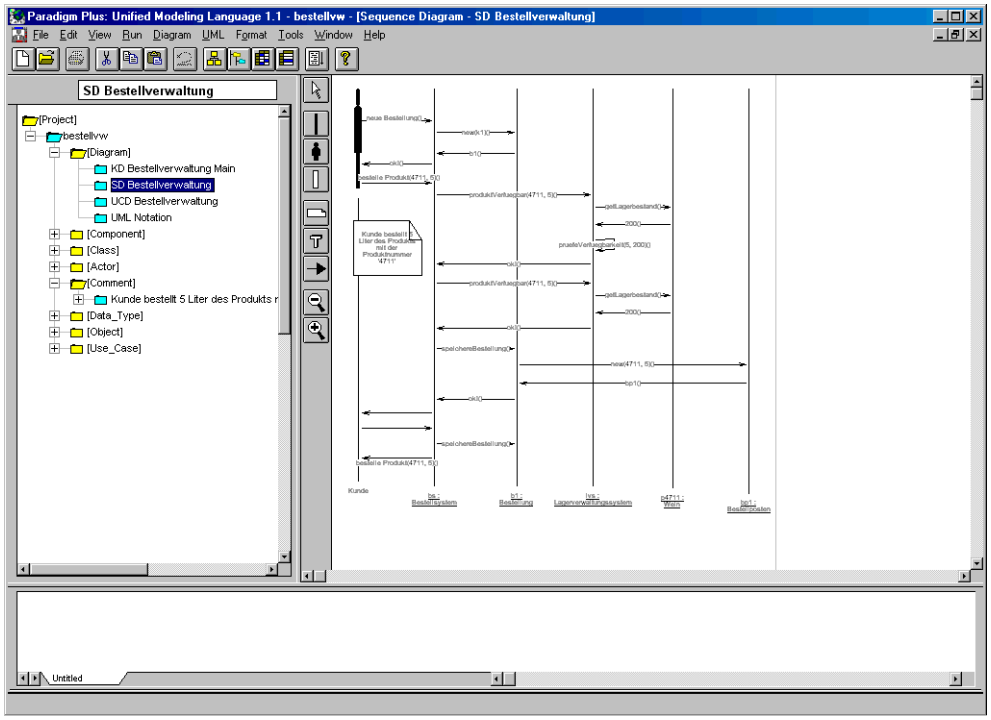

Abb. 6–18
Sequenzdiagramm in Paradigm Plus

Schritt	Vorgangsweise	Benötigte Aktionen
05 Anlegen eines Sequenzdiagramms für den Use Case *Bestellungen verwalten*	RK auf den UseCase *Bestellungen verwalten* im Diagrammfenster. Auswahl des Punktes *New Child* im Kontextmenü. Auswahl des Typs *Sequence Diagram* aus einer Drop-Down Liste und Eingabe des Diagrammnamens im Textfeld. Bestätigen mit *ok*. Es folgt ein weiterer Dialog zur Abfrage des Diagrammtyps und -namens, der mit *ok* bestätigt werden muß.	*Maus:* 7 *Tastatur:* 1 *8*
06 Anlegen von einem Aktor und 5 "leeren" Objekten	Auswählen des *Actor* bzw. *Object* Werkzeugs und Anlegen der Elemente im Diagramm. Beenden durch LK auf das Pfeilsymbol.	*Maus:* 9 *Tastatur:* 0 *9*
07 Einzeichnen des Aktors Kunde in das Sequenzdiagramm	DK auf den Aktor, um dessen Eigenschafts-Dialog zu öffnen. Auswahl des Bezeichners *Kunde* in der Drop-Down Liste *Name*. Schließen des Eigenschafts-Dialoges mit *ok*.	*Maus:* 4 *Tastatur:* 0 *4*
08 Anlegen der Objekte Wein, Bestellposten und Bestellung	Öffnen der Eigenschafts-Dialoge der entsprechenden Objekte mittels DK. Eingabe eines Bezeichners, Festlegen des Typs durch Auswahl der Klasse in der Liste *Class* und Bestätigen der Aktion mit der Schaltfläche *ok*.	*Maus:* 12 *Tastatur:* 3 *15*
09 Anlegen der Objekte Bestellsystem und Lagerverwaltungssystem.	Analog Punkt 07, Paradigm Plus fordert auch die Eingabe eines Objektbezeichners. Weiters müssen noch vor Einzeichnen der Nachrichten alle Lebenslinien der Objekte manuell verlängert werden (LK und Drag&Drop), da das Tool diese nicht automatisch anpaßt.	*Maus:* 20 *Tastatur:* 2 *22*
10 Anlegen der Messages *neue Bestellung* und *new*, inklusive der Rückgabenachrichten	Auswählen des *Relationship*-Werkzeugs und Anlegen der vier Kanten im Diagramm. Bei Beziehungen zwischen Objekten muß noch im aufscheinenden Dialogfenster der Typ (*Message*, *Link*, *Depends_on*) ausgewählt werden.	*Maus:* 11 *Tastatur:* 0 *11*
11 Anlegen nächsten 6 Messages (von *bestelle Produkt (4711, 5)* bis zur dazugehörigen Rückgabenachricht)	Analog Punkt 09. Bei der rekursiven Nachricht muß zweimal in die Diagrammfläche geklickt werden, um die gewünschte Linienführung zu erhalten.	*Maus:* 19 *Tastatur:* 0 *19*

Schritt	Vorgangsweise	*Benötigte Aktionen*
12 Anlegen der 4 Messages (von *aktualisiereLagerbestand(4711, 5)* bis zur dazugehörigen Rückgabenachricht)	Analog Punkt 09.	*Maus: 12* *Tastatur: 0* *12*
13 Anlegen der verbleibenden 8 Messages	Analog Punkt 09.	*Maus: 24* *Tastatur: 0* *24*
14 Vergeben der Nachrichten-Bezeichner	Eigenschafts-Dialoge der Nachrichten mittels DK aufrufen, einen Bezeichner eingeben und mit Enter bestätigen.	*Maus: 22* *Tastatur: 22* *44*
15 Anlegen der Notiz	Auswählen des *Comment*-Werkzeugs aus der Werkzeugleiste, Anlegen des Textfeldes im Diagramm, LK auf Pfeilsymbol, DK auf die Notiz und Eingabe des. **Die Notiz kann nicht mit der Nachricht verbunden werden.**	*Maus: 6* *Tastatur: 1* *7*
Gesamt		***210***

6.4.7 Use-Case-/Sequenzdiagramm - Rational Rose

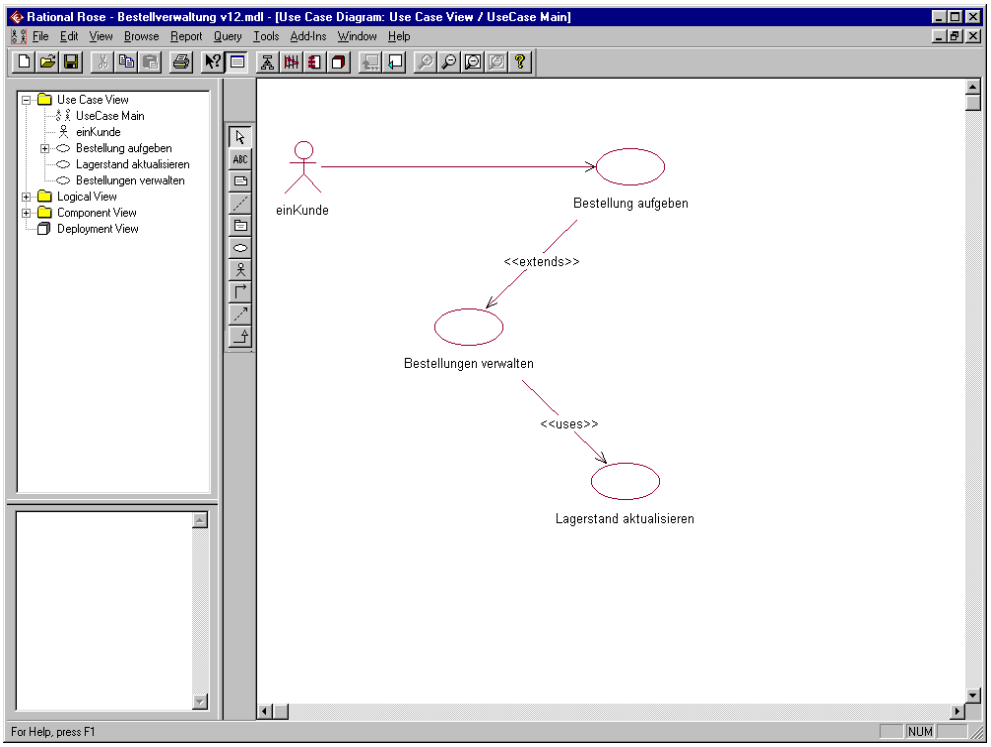

Abb. 6–19
Use-Case Diagramm in Rational Rose

Schritt	Vorgangsweise	*Benötigte Aktionen*
01 Anlegen eines UseCase Diagramms	RK auf den Knoten *UseCase View* im Browser. Auswahl von *New - Use Case Diagram* aus dem Kontextmenü. Eingabe eines Bezeichners und Öffnen des neuen Diagramms.	*Maus: 3* *Tastatur: 1* *4*
02 Anlegen des Aktors Kunde	Auswahl des Werkzeugs *Actor*, LK in den Zeichenbereich und Eingabe eines Namens für den neuen Aktor.	*Maus: 2* *Tastatur: 1* *3*
03 Anlegen der drei Use Cases	Auswahl des *Use Case* Werkzeugs aus der Werkzeugleiste. Anlegen des ersten Use Cases mit einem Shift-LK auf die Zeichenfläche und Eingabe des Bezeichners. Anlegen der beiden anderen Use Cases auf die gleiche Art.	*Maus: 4* *Tastatur: 3* *7*
04 Anlegen der Assoziationen zwischen den Use Cases	LK auf das *Association*-Tool in der Werkzeugleiste. Anlegen aller drei Assoziationen hintereinander mit gedrückt gehaltener Shift-Taste. DK auf die Assoziation zwischen *Bestellung aufgeben* und *Bestellungen verwalten* und Einstellen des Stereotyps *extends* auf der ersten Dialogseite des Eigenschaften-Dialoges der Beziehung. Ebenso Einstellen des Stereotyps *uses* für die Assoziation zwischen *Bestellungen verwalten* und *Lagerstand aktualisieren*.	*Maus: 12* *Tastatur: 0* *12*

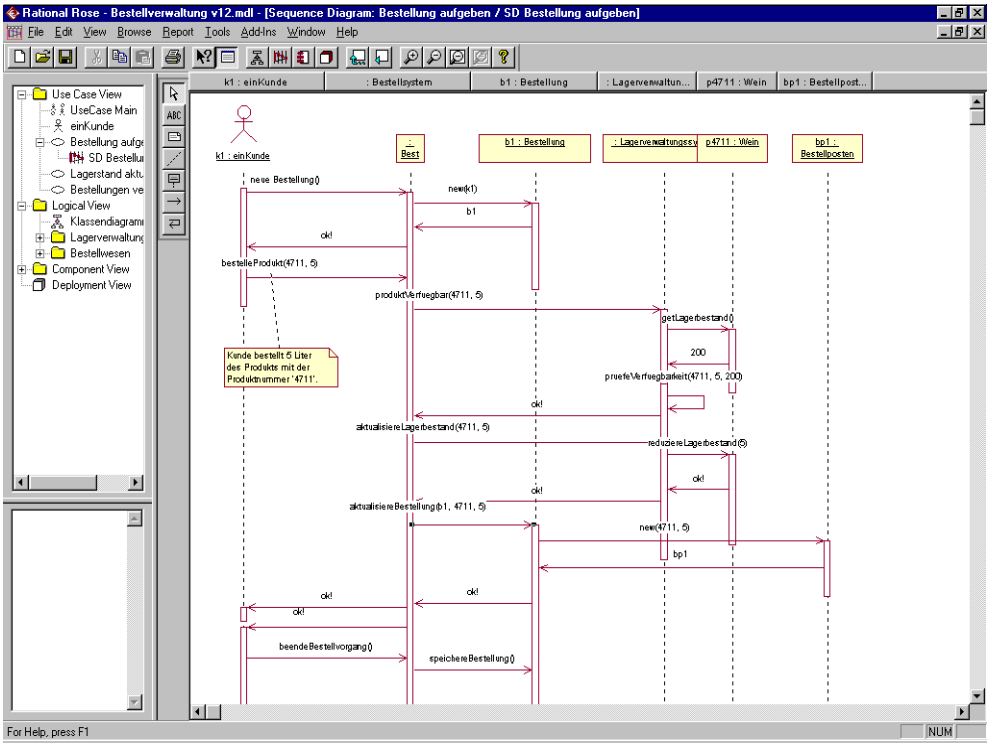

Abb. 6–20
Sequenzdiagramm in Rational Rose

Schritt	Vorgangsweise	Benötigte Aktionen
05 Anlegen eines Sequenzdiagramms für den Use Case *Bestellungen verwalten*	RK auf den UseCase *Bestellungen verwalten* im Browser. Auswahl von *New - Sequence Diagram* aus dem Kontextmenü. Eingabe eines Bezeichners und Öffnen des neuen Diagramms.	*Maus:* 3 *Tastatur:* 1 *4*
06 Anlegen von 6 "leeren" Objekten	Auswählen des *Object* Werkzeugs und Anlegen der Objekte im Diagramm mit Shift-LK.	*Maus:* 7 *Tastatur:* 0 *7*
07 Einzeichnen des Aktors Kunde in das Sequenzdiagramm	DK auf das erste Objekt, um dessen Eigenschafts-Dialog zu öffnen. Eingabe eines Bezeichners, danach Festlegen des Typs des Objekts auf *einKunde* (zuvor im Use-Case Diagramm angelegter Aktor). Schließen des Eigenschafts-Dialoges mit *ok*.	*Maus:* 5 *Tastatur:* 1 *6*
08 Anlegen der Objekte Wein, Bestellposten und Bestellung	Öffnen der Eigenschafts-Dialoge der entsprechenden Objekte mittels DK. Eingabe eines Bezeichners, Festlegen des Typs und Bestätigen der Aktion mit der Schaltfläche *ok*.	*Maus:* 12 *Tastatur:* 3 *15*

Schritt	Vorgangsweise	*Benötigte Aktionen*
09 Anlegen der Objekte Bestellsystem und Lagerverwaltungssystem.	Analog Punkt 08, jedoch ohne die Eingabe eines Bezeichners für die beiden Objekte.	*Maus: 8* *Tastatur: 0* *8*
10 Anlegen der Messages *neue Bestellung* und *new*, inklusive der Rückgabenachrichten	Auswählen des *Object Message*-Werkzeugs und Anlegen der vier Kanten im Diagramm durch Shift-Drag&Drop. Danach einzeln die Eigenschafts-Dialoge der Nachrichten mittels DK aufrufen, einen Bezeichner eingeben und mit Enter bestätigen.	*Maus: 9* *Tastatur: 4* *13*
11 Anlegen nächsten 6 Messages (von *bestelle Produkt (4711, 5)* bis zur dazugehörigen Rückgabenachricht)	Analog Punkt 10. Bei der rekursiven Nachricht muß das Werkzeug *Message to self* verwendet werden.	*Maus: 14* *Tastatur: 6* *20*
12 Anlegen der 4 Messages (von *aktualisiereLagerbestand(4711, 5)* bis zur dazugehörigen Rückgabenachricht)	Analog Punkt 10.	*Maus: 9* *Tastatur: 4* *13*
13 Anlegen der verbleibenden 8 Messages	Analog Punkt 10.	*Maus: 17* *Tastatur: 8* *25*
14 Anlegen der Notiz	Auswählen des *Note*-Werkzeugs aus der Werkzeugleiste, Anlegen des Textfeldes im Diagramm und Eingabe des Textes. Vergrößern des Textfeldes per Drag&Drop, damit der Text vollständig sichtbar wird. Auswählen des *Note Connector*-Werkzeugs und Verbinden der Klasse Bestellung mit der Notiz.	*Maus: 5* *Tastatur: 1* *6*
Gesamt		***143***

6.4.8 Use-Case-/Sequenzdiagramm - SELECT Enterprise

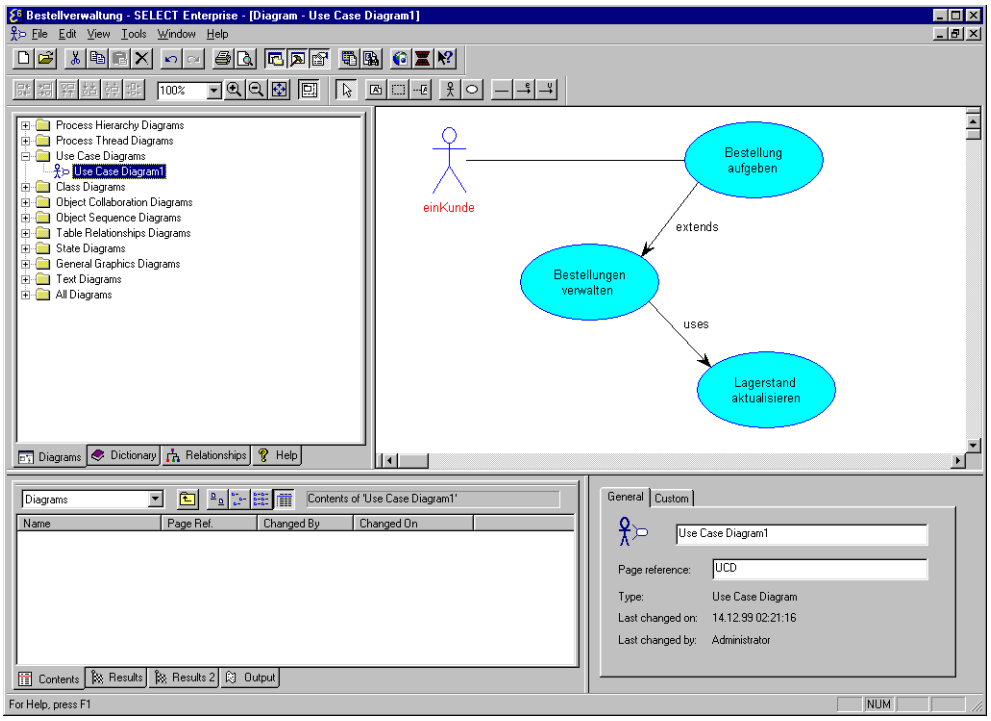

Abb. 6–21
Use-Case Diagramm in SELECT Enterprise

Schritt	Vorgangsweise	*Benötigte Aktionen*
01 Anlegen eines UseCase Diagramms	RK auf den Knoten *UseCase Diagram* im Browser. Auswahl von *New Use Case Diagram* aus dem Kontextmenü. Das Diagramm wird angelegt.	*Maus:* 2 *Tastatur:* 0 2
02 Anlegen des Aktors Kunde	Auswahl des Werkzeugs *Actor*, LK in den Zeichenbereich und Eingabe eines Namens für den neuen Aktor.	*Maus:* 2 *Tastatur:* 1 3
03 Anlegen der drei Use Cases	Auswahl des *Use Case* Werkzeugs aus der Werkzeugleiste. Anlegen des ersten Use Cases mit einem LK auf die Zeichenfläche und Eingabe des Bezeichners. Anlegen der beiden anderen Use Cases auf die gleiche Art.	*Maus:* 6 *Tastatur:* 3 9
04 Anlegen der Assoziationen zwischen den Use Cases	Auswahl des entsprechenden Tools in der Werkzeugleiste (*Interaction* für die Beziehung zwischen Aktor und Use Case *Bestellung aufgeben*, *Extends Flow* für *Bestellung aufgeben* und *Bestellungen verwalten* und *Uses Flow* für die Beziehung zwischen *Bestellungen verwalten* und *Lagerstand aktualisieren*).	*Maus:* 9 *Tastatur:* 0 9

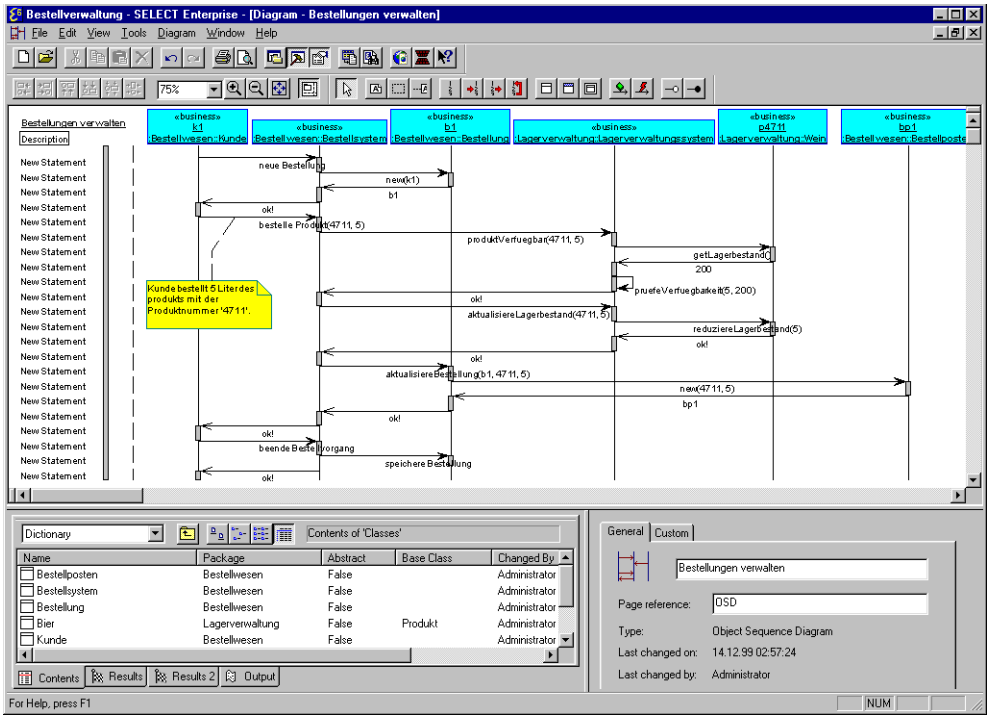

Abb. 6–22
Sequenzdiagramm in SELECT Enterprise

Schritt	Vorgangsweise	Benötigte Aktionen
05 Anlegen eines Sequenzdiagramms für den Use Case *Bestellungen verwalten*	RK auf den UseCase *Bestellungen verwalten* im Browser. Auswahl von *New - Object Sequence Diagram* aus dem Kontextmenü.	*Maus:* 2 *Tastatur:* 0 2
06 Anlegen der Objekte im Sequenzdiagramm	Um die Zuordnung der Objekte zu den entsprechenden Klassen zu ermöglichen, werden diese nicht direkt in das Sequenzdiagramm eingezeichnet, sondern aus der Datenbasis übernommen: LK auf den Reiter *Dictionary* im Browser Fenster. Öffnen des Knotens *Classes* und Übertragen der Klassen in das Diagrammfenster mittels Drag&Drop. Da kein Aktor in das Sequenzdiagramm übernommen werden kann, wird eine Instanz der Klasse Kunde angelegt.	*Maus:* 8 *Tastatur:* 0 8
07 Vergabe der korrekten Objekt-Bezeichner	Zweimaliger LK (kein DK) in das Objekt und Vergabe des Bezeichners.	*Maus:* 12 *Tastatur:* 6 18

Schritt	Vorgangsweise	*Benötigte Aktionen*
08 Anlegen der Messages *neue Bestellung* und *new*, inklusive der Rückgabenachrichten	Das Eintragen von Nachrichten oder Events gestaltet sich in SELECT Enterprise ein wenig umständlicher als bei anderen Tools: Zuerst muß eine neue Sequenz (Werkzeug *Sequence* in der Toolleiste) angelegt werden. Weiters sollte eine nähere Beschreibung eingetragen werden, welches jedoch hier nicht explizit durchgeführt wird. Danach kann dann die Nachricht eingezeichnet werden. Dazu wird das Werkzeug *Operation* ausgewählt und die Verbindung zwischen den entsprechenden Objekten angelegt. Daraufhin kann im folgenden Dialog eine Operation der Zielklasse ausgewählt, oder eine neue angelegt werden.	*Maus: 28* *Tastatur: 4* *32*
09 Anlegen nächsten 6 Messages (von *bestelle Produkt (4711, 5)* bis zur dazugehörigen Rückgabenachricht)	Analog Punkt 08. Bei der rekursiven Nachricht muß zweimal das selbe Objekt angewählt werden.	*Maus: 43* *Tastatur: 6* *49*
10 Anlegen der 4 Messages (von *aktualisiereLagerbestand(4711, 5)* bis zur dazugehörigen Rückgabenachricht)	Analog Punkt 09.	*Maus: 28* *Tastatur: 4* *32*
11 Anlegen der verbleibenden 8 Messages	Analog Punkt 09.	*Maus: 56* *Tastatur: 8* *64*
12 Anlegen der Notiz	Auswählen des *Note*-Werkzeugs aus der Werkzeugleiste, Anlegen des Textfeldes im Diagramm und Eingabe des Textes. Auswählen des *Link Note*-Werkzeugs und verbinden der Nachricht mit der Notiz.	*Maus: 5* *Tastatur: 1* *6*
Gesamt		***234***

6.4.9 Use-Case-/Sequenzdiagramm - StP/UML

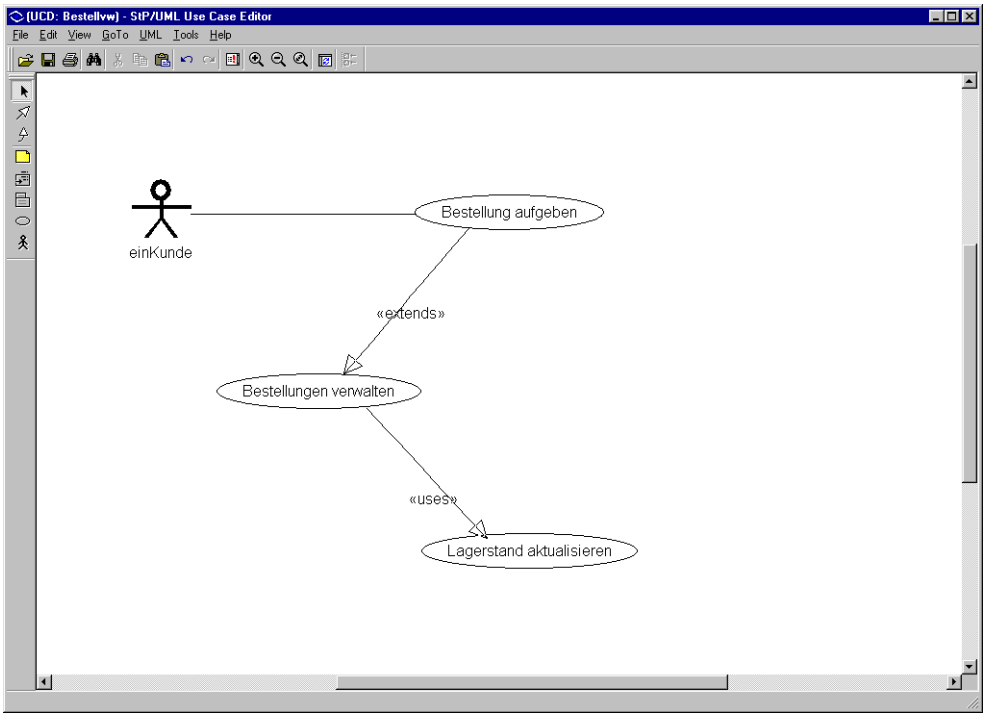

Abb. 6–23
Use-Case Diagramm in StP/UML

Schritt	Vorgangsweise	*Benötigte Aktionen*
01 Anlegen eines UseCase Diagramms	LK im Model Browser auf den Eintrag *Use Case*, RK im *Use Case* Feld. Auswahl von *New* aus dem Kontextmenü. Der Bezeichner wird erst beim Speichern des Diagrammes im entsprechenden Dialog eingegeben.	*Maus: 4* *Tastatur: 1* *5*
02 Anlegen des Aktors Kunde	Auswahl des Werkzeugs *Actor*, LK in den Zeichenbereich und Eingabe eines Namens für den neuen Aktor.	*Maus: 2* *Tastatur: 1* *3*
03 Anlegen der drei Use Cases	Auswahl des *Use Case* Werkzeugs aus der Werkzeugleiste durch DK. Anlegen des ersten Use Cases durch LK auf die Zeichenfläche und Eingabe des Bezeichners. Anlegen der beiden anderen Use Cases auf die gleiche Art. Beenden durch LK auf das Pfeilsymbol in der Werkzeugleiste.	*Maus: 5* *Tastatur: 3* *8*
04 Anlegen der Assoziationen zwischen den Use Cases	DK auf das *Arc*-Tool in der Werkzeugleiste. Anlegen aller drei Assoziationen hintereinander. Bei einer Beziehung zwischen zwei Use Cases wird automatisch der Stereotyp *uses* vergeben. Dieser muß für die Beziehung zwischen *Bestellung aufgeben* und *Bestellungen verwalten* noch geändert werden. LK auf die Beziehung, danach RK und Auswahl des Punktes *Replace...* aus dem Kontextmenü. Der Stereotyp wird automatisch auf *extends* geändert.	*Maus: 11* *Tastatur: 0* *11*

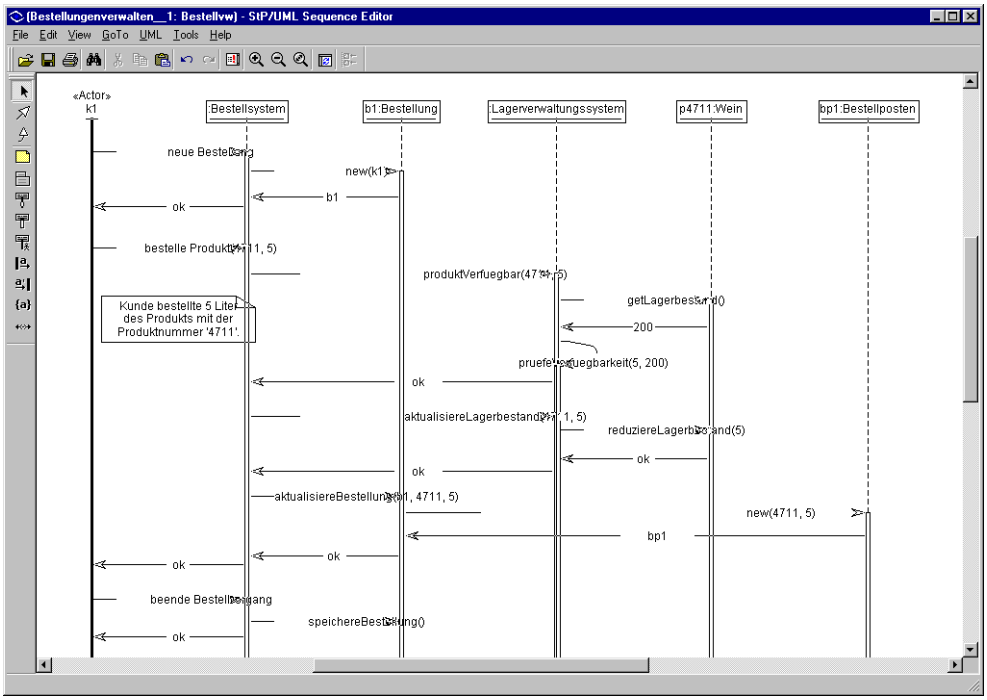

Abb. 6–24
Sequenzdiagramm in StP/UML

Schritt	Vorgangsweise	Benötigte Aktionen
05 Anlegen eines Sequenzdiagramms für den Use Case *Bestellungen verwalten*	LK auf den Use Case *Bestellungen verwalten* im Diagramm. Auswahl des Menüs *GoTo - Scenario in Sequence Diagram*. Ein neues Sequenzdiagramm wird angelegt, wobei ein Bezeichner vergeben werden muß.	*Maus:* 4 *Tastatur:* 1 *5*
06 Einzeichnen des Aktors Kunde in das Sequenzdiagramm	Auswählen des *Actor*-Werkzeugs und Anlegen des Aktors durch LK im Diagramm. Anschließend Vergabe des Bezeichners.	*Maus:* 2 *Tastatur:* 1 *3*
07 Anlegen der Objekte	DK auf das *Passive Object*-Werkzeug und Anlegen aller Objekte im Diagramm. Gleichzeitig Vergabe des Bezeichners und Typs in der Form *Objektname:Klassenname*.	*Maus:* 7 *Tastatur:* 5 *12*
08 Anlegen der Messages *neue Bestellung* und *new*, inklusive der Rückgabenachrichten	Auswählen des *Arc*-Werkzeugs durch DK und Anlegen der vier Kanten im Diagramm. Danach DK auf die entsprechende Nachricht und Eingabe des Bezeichners. Anschließend muß im Menü *UML* der Punkt *Expand All Objects 50%* ausgewählt werden, um die Lebenslinien der Objekte für weitere Diagrammeinträge zu verlängern. Dies muß im Folgenden mehrmals durchgeführt werden, daher wird hier eine dreimalige Auswahl gezählt.	*Maus:* 20 *Tastatur:* 4 *24*

Schritt	Vorgangsweise	Benötigte Aktionen
09 Anlegen nächsten 6 Messages (von *bestelle Produkt (4711, 5)* bis zur dazugehörigen Rückgabenachricht)	Analog Punkt 08. Bei der rekursiven Nachricht muß zweimal in die Diagrammfläche geklickt werden, um die gewünschte Linienführung zu erhalten.	*Maus: 22* *Tastatur: 6* *28*
10 Anlegen der 4 Messages (von *aktualisiereLagerbestand(4711, 5)* bis zur dazugehörigen Rückgabenachricht)	Analog Punkt 09.	*Maus: 10* *Tastatur: 4* *14*
11 Anlegen der verbleibenden 8 Messages	Analog Punkt 09.	*Maus: 18* *Tastatur: 8* *26*
12 Anlegen der Notiz	Auswählen des *Note*-Werkzeugs aus der Werkzeugleiste, Anlegen des Textfeldes im Diagramm und Eingabe des Textes. **Die Notiz kann nicht mit der Nachricht verbunden werden.**	*Maus: 2* *Tastatur: 1* *3*
Gesamt		***142***

6.4.10 Use-Case-/Sequenzdiagramm - WithClass

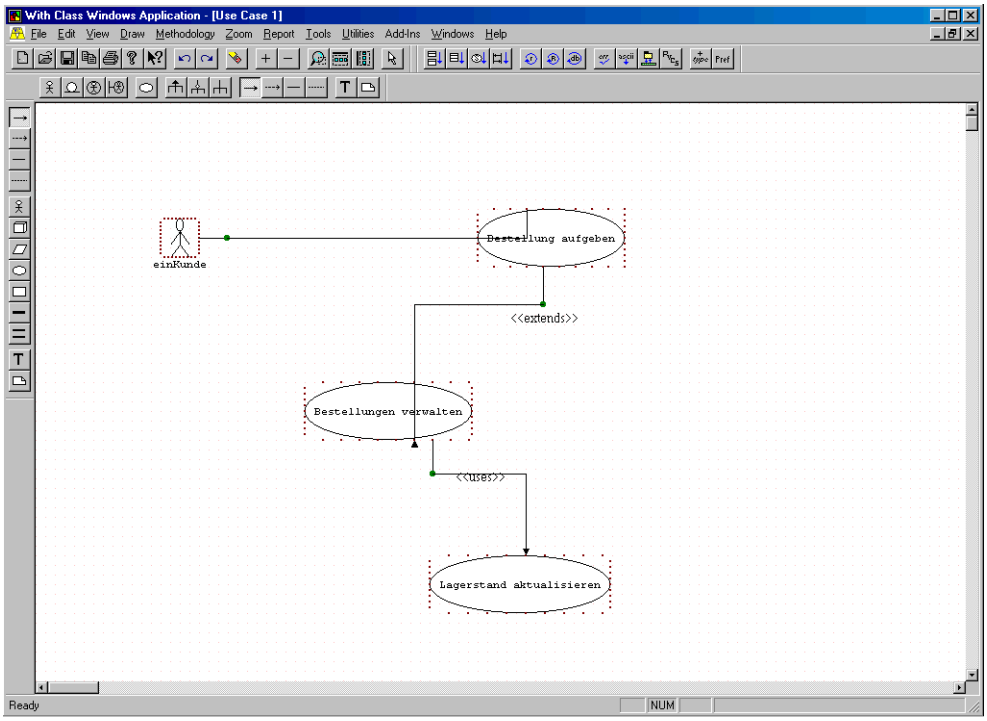

Abb. 6–25
Use-Case Diagramm in WithClass

Schritt	Vorgangsweise	Benötigte Aktionen
01 Anlegen eines UseCase Diagramms	In der *Project Information* RK auf den Eintrag *Use Case* und Auswahl des Punkts *New* im Kontextmenü.	*Maus: 2* *Tastatur: 0* *2*
02 Anlegen des Aktors Kunde	Auswahl des Werkzeugs *Actor Shape*, LK in den Zeichenbereich und Eingabe eines Namens für den neuen Aktor.	*Maus: 2* *Tastatur: 1* *3*
03 Anlegen der drei Use Cases	Auswahl des *Use Case* Werkzeugs aus der Werkzeugleiste. Anlegen des Use Case mit einem LK auf die Zeichenfläche und Eingabe des Bezeichners. Anlegen der beiden anderen Use Cases auf die gleiche Art.	*Maus: 6* *Tastatur: 3* *9*
04 Anlegen der Assoziationen zwischen den Use Cases	LK auf das entsprechende-Tool (*Line Shape* bzw. *Arrow*) in der Werkzeugleiste. Anlegen der Assoziationen. DK auf die Assoziation zwischen *Bestellung aufgeben* und *Bestellungen verwalten* und Eingabe des Stereotyps *extends* auf der ersten Dialogseite des Eigenschaften-Dialoges der Beziehung. Ebenso Einstellen des Stereotyps *uses* für die Assoziation zwischen *Bestellungen verwalten* und *Lagerstand aktualisieren*.	*Maus: 13* *Tastatur: 2* *15*

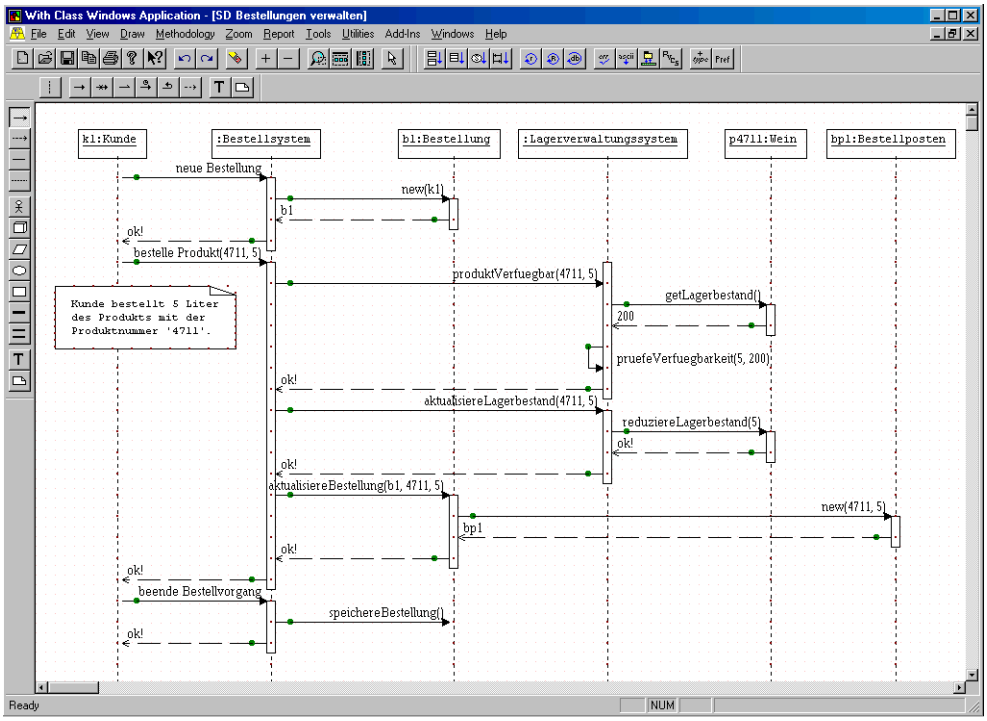

Abb. 6–26
Sequenzdiagramm in WithClass

Schritt	Vorgangsweise	Benötigte Aktionen
05 Anlegen eines Sequenzdiagramms für den Use Case *Bestellungen verwalten*	In der *Project Information* RK auf den Eintrag *Sequence* und Auswahl des Punkts *New* im Kontextmenü, danach Eingabe eines Diagrammtitels. Das Sequenzdiagramm kann nicht einem bestimmten Use Case zugeordnet werden.	*Maus: 3* *Tastatur: 1* *4*
06 Anlegen von 6 "leeren" Objekten	Auswählen des *Object* Werkzeugs, Anlegen des Objekts im Diagramm mit LK und Vergabe des Bezeichners, wo gewünscht. Wiederholen für die restlichen Objekte.	*Maus: 12* *Tastatur: 4* *16*
07 Anlegen der Klassenzuordnungen	Öffnen der Eigenschafts-Dialoge der entsprechenden Objekte mittels DK. Festlegen der Klasse in der Drop-Down Liste *Class Name* und bestätigen der Aktion mit der Schaltfläche *ok*. Da kein Aktor angelegt werden kann, wird der vorhandene Aktor als Instanz der Klasse Kunde modelliert.	*Maus: 24* *Tastatur: 0* *24*

Schritt	Vorgangsweise	Benötigte Aktionen
08 Anlegen der Messages *neue Bestellung* und *new*, inklusive der Rückgabenachrichten	Auswählen des *Call*-Werkzeugs (bzw. *Interaction Return* bei Rückgabe eines Wertes) und Anlegen der Nachrichten im Diagramm durch LK in die Lebenslinie des Quellobjekts, danach LK in die Lebenslinie des Zielobjekts. Im Folgenden Eingabe des Bezeichners. Das Tool fragt nun ab, ob die Nachricht als neue Operation im Klassendiagramm angelegt werden soll, welches bestätigt, oder abgelehnt werden kann. Leider wird dies jedesmal beim Anlegen einer neuen Nachricht für jede bereits angelegte Nachricht erneut abgefragt, gleich, ob die übrigen Nachrichten bereits im Klassendiagramm angelegt worden sind, oder nicht. Als Resultat muß der Benutzer etwa beim Anlegen der zehnten Nachricht auch zehnmal den selben Dialog bestätigen. Da es sich hierbei offensichtlich um einen Fehler des Programms handelt, wird dies bei der empirischen Messung nicht berücksichtigt, da sich sonst Verzerrungen der Ergebnisse ergeben könnten.	*Maus:* 12 *Tastatur:* 4 16
09 Anlegen nächsten 6 Messages (von *bestelle Produkt (4711, 5)* bis zur dazugehörigen Rückgabenachricht)	Analog Punkt 08. Bei der rekursiven Nachricht erfolgt ein doppelter LK in das selbe Objekt.	*Maus:* 19 *Tastatur:* 6 25
10 Anlegen der 4 Messages (von *aktualisiereLagerbestand(4711, 5)* bis zur dazugehörigen Rückgabenachricht)	Analog Punkt 09.	*Maus:* 12 *Tastatur:* 4 16
11 Anlegen der verbleibenden 8 Messages	Analog Punkt 09.	*Maus:* 24 *Tastatur:* 8 32

Schritt	Vorgangsweise	Benötigte Aktionen
12 Anlegen der Notiz	Auswählen des *Note Shape*-Werkzeugs aus der Werkzeugleiste, Anlegen des Textfeldes im Diagramm und Eingabe des Textes. Vergrößern des Textfeldes per Drag&Drop, damit der Text vollständig sichtbar wird. **Die Notiz kann nicht mit einer Nachricht verbunden werden.**	*Maus: 4* *Tastatur: 1* *5*
Gesamt		***167***

6.5 Erstellung des Zustandsdiagramms

6.5.1 Zusammenfassung der Ergebnisse

In Abb. 6–27 können die Ergebnisse der Analyse der Erstellung des Zustandsdiagramms abgelesen werden. Das Programm objectiF wurde bei diesem Punkt nicht bewertet, da es nur einfachste Zustandsdiagramme darstellen kann, und ein Vergleich mit der übrigen Tools somit wenig sinnvoll erschien. Ebenfalls nicht in die Wertung aufgenommen wurde OTW-2, da das Werkzeug keine UML-konformen Zustandsdiagramme unterstützt und statt dessen Objekt-Prozeß-Diagramme verwendet.

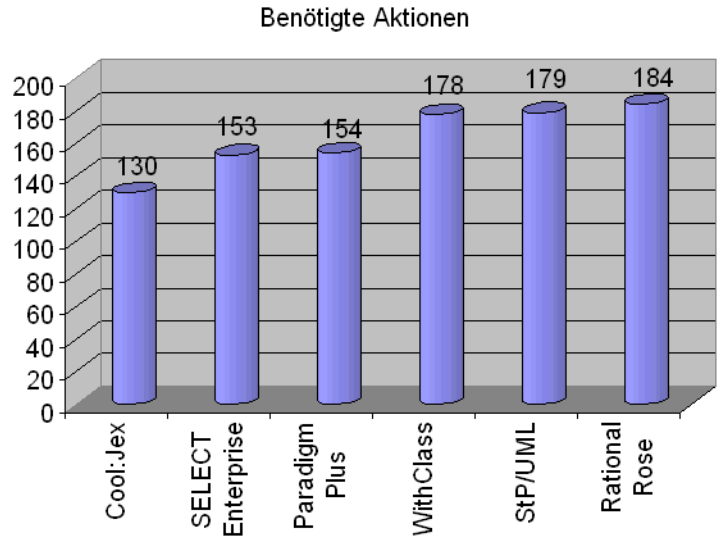

Abb. 6–27
Ergebnisse der Zustandsdiagrammerstellung

6.5.2 Interpretation der Ergebnisse

Bei der Konstruktion des Zustandsdiagrammes liegen die Ergebnisse der Kandidaten näher beieinander als bei den anderen Tests. Wieder ist es so, daß die Möglichkeit, Parameter als String direkt in das Diagramm eintragen zu können, die Arbeit beschleunigt.

Cool:Jex ist das einzige Programm, das dieses Konzept konsequent umsetzt. Es erlaubt alle Informationen - sowohl der Zustände, als auch der Zustandsübergänge - textuell in das Diagramm zu schreiben. Bei allen anderen Werkzeugen ist der Benutzer gezwungen, zumindest teilweise mit Dialogen oder Tabellen zu arbeiten.

6.5.3 Zustandsdiagramm - Cool:Jex

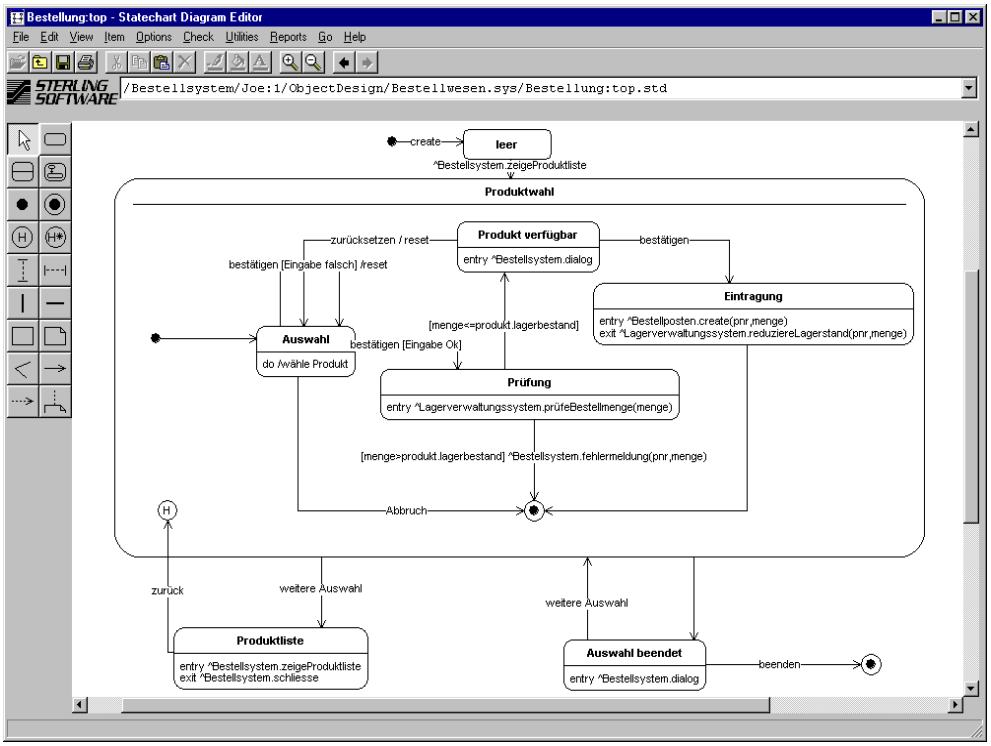

Abb. 6–28
Zustandsdiagramm in Cool:Jex

Schritt	Vorgangsweise
01 Anlegen eines Zustandsdiagramms für die Klasse Bestellung	DK auf die Klasse Bestellung im Klassendiagramm. DK im Dialog *Select Item* auf den Namen der Klasse und Generieren eines neuen Zustandsdiagramms im darauffolgenden Dialog.

Benötigte Aktionen

Maus:	*3*
Tastatur:	*0*
	3

Schritt	Vorgangsweise	Benötigte Aktionen
02 Anlegen von Start-, End-, eines Super- und von 3 "normalen" Zuständen	Auswahl des Werkzeugs *Start State* und LK in den Zeichenbereich des Diagramms um den Startzustand zu plazieren. Um einen Endzustand anzulegen wird die gleiche Aktion mit dem Werkzeug *End State* durchgeführt. Mit dem *State*- bzw. *Super State*- Werkzeug werden die Zustände in das Diagramm eingezeichnet.	*Maus:* 11 *Tastatur:* 0 *11*
03 Vergabe der Namen für die 4 Zustände	Anwählen des jeweiligen Zustandes mit LK und Eingabe des Bezeichners.	*Maus:* 4 *Tastatur:* 4 *8*
04 Bearbeiten der 4 Zustände	Den Zuständen *Produktliste* und *Auswahl beendet* werden interne Aktionen zugefügt. Dies geschieht ausschließlich durch textuelles Eintragen dieser direkt im Diagramm. Interne Aktionen werden in der unteren Hälfte eines Zustandes in der Form < *Ereignis [Bedingung] /Aktivität ^Empfänger.Nachricht* > angegeben. Einfügen eines History-Zustandes in den Superzustand Produktwahl.	*Maus:* 11 *Tastatur:* 2 *13*
05 Anlegen von Start-, End- und von 4 "leeren" Zuständen als Subzustände des Zustandes *Produktwahl*	Analog Punkt 02, jedoch werden alle Elemente innerhalb des Zustandes *Produktwahl* angelegt.	*Maus:* 9 *Tastatur:* 0 *9*
06 Vergabe der Namen für die 4 Subzustände	Analog Punkt 03	*Maus:* 4 *Tastatur:* 4 *8*
07 Bearbeiten der 4 Subzustände	Analog Punkt 04	*Maus:* 4 *Tastatur:* 18 *22*
08 Anlegen der Zustandsübergänge, die mit dem Zustand *leer* in Verbindung stehen	Auswählen des *Transition*- Werkzeugs und Anlegen der Kante im Diagramm durch LK zuerst auf den Ausgangs- und danach auf den Zielzustand. Nach der Konstruktion der Kante kann diese mit Text versehen werden. Hierbei werden auch alle Parameter des Zustandsüberganges in der Form < *Ereignis [Bedingung] /Aktivität ^Empfänger.Nachricht* > festgelegt.	*Maus:* 5 *Tastatur:* 3 *8*

Schritt	Vorgangsweise	Benötigte Aktionen
09 Anlegen der noch nicht vorhandenen Zustandsübergänge, die mit dem Zustand *Produktwahl* in Verbindung stehen	Analog Punkt 08	*Maus:* 10 *Tastatur:* 4 14
10 Anlegen der Zustandsübergänge, die mit dem Zustand *Auswahl* in Verbindung stehen	Analog Punkt 08	*Maus:* 12 *Tastatur:* 8 20
11 Anlegen der übrigen Zustandsübergänge	Analog Punkt 08	*Maus:* 8 *Tastatur:* 6 14
Gesamt		***130***

6.5.4 Zustandsdiagramm - Paradigm Plus

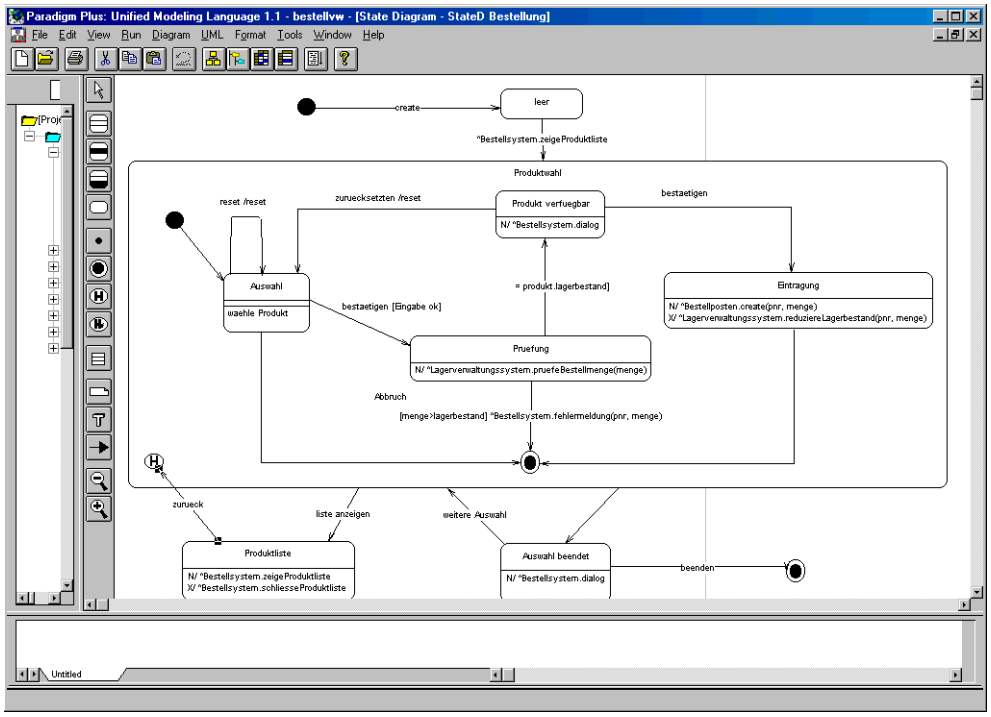

Abb. 6–29
Zustandsdiagramm in Paradigm Plus

Schritt	Vorgangsweise	Benötigte Aktionen
01 Anlegen eines neuen Zustandsdiagramms für die Klasse Bestellung	RK auf die Klasse Bestellung im Klassendiagramm, Auswahl des Menüpunktes *New Child*. Ein Dialog fragt ab, welcher Diagrammtyp erstellt werden soll. Auswahl aus einer Dropdown-Liste und Eingabe eines Diagrammbezeichners. Bestätigung durch LK auf *Attach* und nochmalige Bestätigung zum Öffnen des Diagramms.	*Maus:* 7 *Tastatur:* 1 *8*
02 Anlegen von Start-, End-, eines Super-, und drei "normaler" Zustände mit Bezeichnern	Auswahl folgender Werkzeuge und LK in die Diagrammfläche, um den entsprechenden Zustand anzulegen: ❑ *Initial* ❑ *Final* ❑ *State* Das Tool unterscheidet beim Anlegen nicht zwischen Super- und Subzuständen. Mehrere gleichartige Zustände können automatisch nacheinander eingetragen werden. Um diese Funktion wieder auszuschalten, wird das Pfeilsymbol in der Werkzeugleiste angewählt. Durch DK auf den entsprechenden Zustand wird dessen Eigenschaften-Dialog geöffnet, in dem im Textfeld *Name:* der Bezeichner eingetragen wird.	*Maus:* 14 *Tastatur:* 4 *18*
03 Bearbeiten der Zustände	Um einen Zustand näher zu beschreiben, d.h. interne Aktionen u.a. einzutragen, muß das betreffende Werkzeug (*Action Sequence*, bzw. *Action*) ausgewählt werden. Die *Aktion* wird dann im Zustand abgelegt und näher spezifiziert. Die Angaben *entry:* und *exit:* müssen nicht eingetragen werden, jedoch vergibt das Tool defaultmäßig das Label *N/* für *entry*. Um dies zu ändern, muß der Eigenschaftendialog der Aktion aufgerufen werden (RK, Eintrag *Properties* im Kontextmenü). Der Typ kann nun in einer Dropdown-Liste geändert werden.	*Maus:* 10 *Tastatur:* 3 *13*
04 Anlegen der Subzustände von *Produktwahl*	Vorher muß der Superzustand mittels Drag&Drop vergrößert werden, um die anderen Zustände aufnehmen zu können. Danach DK auf das Werkzeug *State* und Plazieren der Subzustände im Zustand *Produktwahl*, außerdem Eingabe der Bezeichner im Eigenschaftendialog. Ebenso Anlegen von Start- und Endzustand, sowie eines History-Zustands (Werkzeug *Shallow History*).	*Maus:* 16 *Tastatur:* 4 *20*
05 Bearbeiten der Subzustände	Analog Punkt 03	*Maus:* 11 *Tastatur:* 5 *16*

Schritt	Vorgangsweise	Benötigte Aktionen
06 Anlegen der Zustandsübergänge, die mit dem Zustand *leer* in Verbindung stehen	LK auf das Werkzeug *Relationship* und Verbinden der Start- mit den Zielzuständen. Nach Anlegen der Kanten folgt der Eintrag der Parameter des Zustandsübergangs: Deselektion des Werkzeugs, zweimaliger LK (kein DK) auf die entsprechende Kante und Eintrag der Parameter als Freitext-Bezeichner.	*Maus:* 10 *Tastatur:* 2 12
07 Anlegen der noch nicht vorhandenen Zustandsübergänge, die mit dem Zustand *Produktwahl* in Verbindung stehen	Analog Punkt 06	*Maus:* 16 *Tastatur:* 3 19
08 Anlegen der Zustandsübergänge, die mit dem Zustand *Auswahl* in Verbindung stehen	Analog Punkt 06, beim Anlegen des rekursiven Zustandsübergangs, muß zweimal in die leere Diagrammfläche geklickt werden.	*Maus:* 20 *Tastatur:* 4 24
09 Anlegen der restlichen Zustandsübergänge	Analog Punkt 06	*Maus:* 20 *Tastatur:* 4 24
Gesamt		***154***

6.5.5 Zustandsdiagramm - Rational Rose

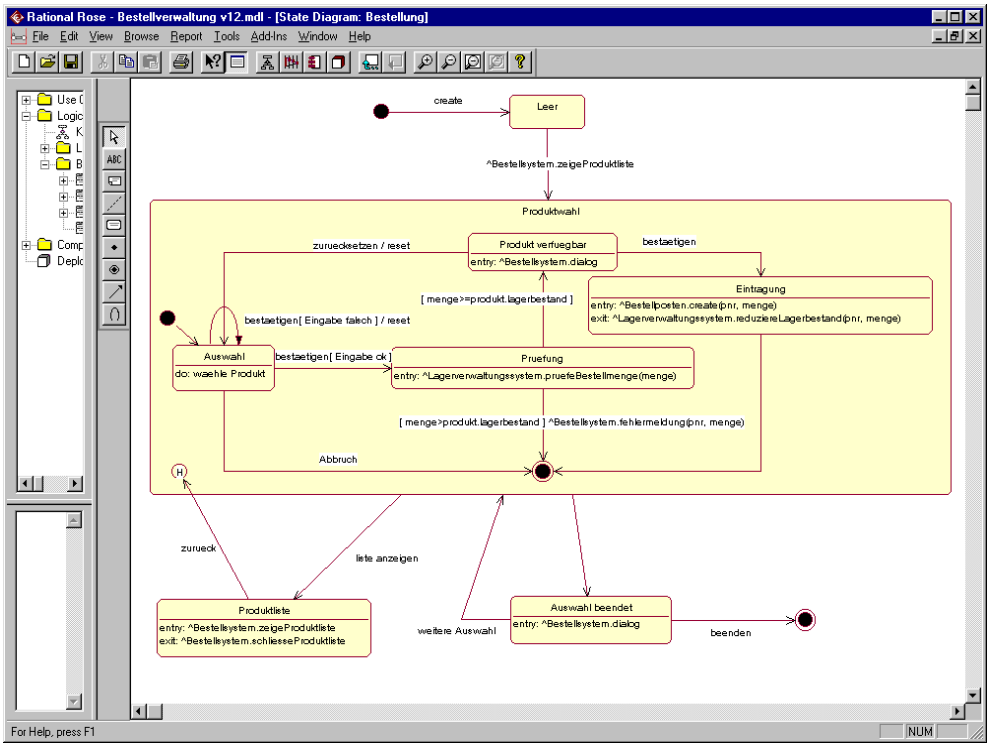

Abb. 6–30
Zustandsdiagramm in Rational Rose

Schritt	Vorgangsweise	Benötigte Aktionen
01 Öffnen des Zustandsdiagramms der Klasse Bestellung	RK auf die Klasse Bestellung im Browser und Öffnen des Zustandsdiagramms über das Kontextmenü. Rational Rose legt für alle Klassen im System automatisch ein leeres Zustandsdiagramm an. Daher muß dieses nicht angelegt, sondern nur geöffnet werden.	*Maus: 2* *Tastatur: 0* *2*
02 Anlegen von Start-, End- und von 4 "leeren" Zuständen	Auswahl des Werkzeugs *Start State* und LK in den Zeichenbereich des Diagramms um den Startzustand zu plazieren. Um einen Endzustand anzulegen wird die gleiche Aktion mit dem Werkzeug *End State* durchgeführt. Mit dem *State*-Werkzeug und vier Shift-LK werden die Zustände in das Diagramm eingezeichnet.	*Maus: 9* *Tastatur: 0* *9*
03 Vergabe der Namen für die 4 Zustände	Anwählen des jeweiligen Zustandes mit LK und Eingabe des Bezeichners.	*Maus: 4* *Tastatur: 4* *8*

Schritt	Vorgangsweise	Benötigte Aktionen
04 Bearbeiten der 4 Zustände	Den Zuständen *Produktliste* und *Auswahl beendet* werden Aktionen zugewiesen. Hierzu wird mit einem DK auf den Zustand sein Eigenschaftsdialog aufgerufen. Auf der Dialogseite *Detail* wird, über ein Kontextmenü, in das Listenfenster *Actions* ein neuer Eintrag eingefügt, und dessen Eigenschaftsdialog mit einem DK geöffnet. In diesem Dialog wird der Typ der Aktion auf *send event* festgelegt und in die jeweiligen Textfelder das auszulösende Ereignis und der Empfänger eingetragen. Der Zustand *leer* wird nicht modifiziert. Der Zustand *Produktwahl* wird per Drag&Drop vergrößert, damit er später Subzustände aufnehmen kann. Weiters erhält er ein History-Icon. Hierzu muß in seinem Eigenschaftsdialog auf der Seite *Detail* die Checkbox *States history* markiert werden.	*Maus:* 33 *Tastatur:* 6 *39*
05 Anlegen von Start-, End- und von 4 "leeren" Zuständen als Subzustände des Zustandes *Produktwahl*	Analog Punkt 02, jedoch werden alle Elemente innerhalb des Zustandes *Produktwahl* angelegt.	*Maus:* 9 *Tastatur:* 0 *9*
06 Vergabe der Namen für die 4 Subzustände	Analog Punkt 03	*Maus:* 4 *Tastatur:* 4 *8*
07 Bearbeiten der 4 Subzustände	Analog Punkt 04	*Maus:* 50 *Tastatur:* 12 *62*
08 Anlegen der Zustandsübergänge, die mit dem Zustand *leer* in Verbindung stehen	Auswählen des *State Transition*- Werkzeugs und Anlegen der Kante im Diagramm durch Ziehen mit der Maus. Nach der Konstruktion der Kante kann diese sofort benannt werden. Mit dem Bezeichner können auch die meisten anderen Parameter, statt im Eigenschaftendialog, direkt in das Diagramm eingetragen werden. Nach dem Zeichen ^ folgt ein Empfängerobjekt, nach einem weiteren Punkt das zu sendende Ereignis und eventuelle Parameter werden eingeklammert notiert. In eckigen Klammern werden Bedingungen dargestellt und nach einem Schrägstrich Aktionen.	*Maus:* 4 *Tastatur:* 3 *7*

Schritt	Vorgangsweise	Benötigte Aktionen
09 Anlegen der noch nicht vorhandenen Zustandsübergänge, die mit dem Zustand *Produktwahl* in Verbindung stehen	Analog Punkt 08	*Maus: 3* *Tastatur: 1* *4*
10 Anlegen der Zustandsübergänge, die mit dem Zustand *Auswahl* in Verbindung stehen	Analog Punkt 08	*Maus: 10* *Tastatur: 8* *18*
11 Anlegen der übrigen Zustandsübergänge	Analog Punkt 08	*Maus: 8* *Tastatur: 10* *18*
Gesamt		***184***

6.5.6 Zustandsdiagramm - SELECT Enterprise

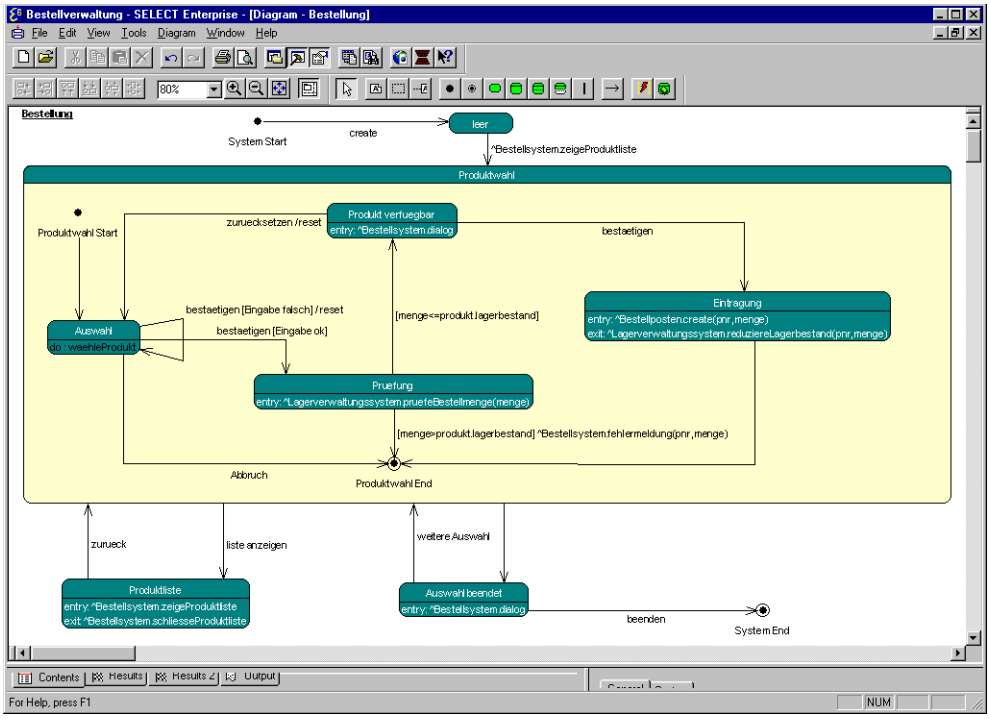

Abb. 6–31
Zustandsdiagramm in SELECT Enterprise

Schritt	Vorgangsweise	*Benötigte Aktionen*
01 Anlegen eines neuen Zustandsdiagramms für die Klasse Bestellung	RK auf die Klasse Bestellung im Klassendiagramm, Auswahl des Punktes *New - State Diagram* im Kontextmenü.	*Maus: 2* *Tastatur: 0* *2*
02 Anlegen von Start-, End-, eines Super-, und drei "normaler" Zustände mit Bezeichnern	Auswahl folgender Werkzeuge und LK in die Diagrammfläche, um den entsprechenden Zustand anzulegen: ❑ *Start State* ❑ *Result State* ❑ *Sequential State* ❑ *Atomic State* Um mehrere gleichartige Zustände nacheinander einzutragen, könnten diese mittels gedrückter STRG-Taste abgelegt werden, diese Funktion wird jedoch nicht verwendet, da der Bezeichner erst nach dem Anlegen aller Zustände extra vergeben werden müßte. Daher werden die Bezeichner der Zustände werden sofort nach Anlegen derselben eingetragen, auch für Start- und Endzustände müssen Bezeichner vergeben werden.	*Maus: 12* *Tastatur: 6* *18*
03 Bearbeiten der Zustände	Um einen Zustand näher zu beschreiben, d.h. interne Aktionen u.a. einzutragen, muß das betreffende Werkzeug (*Event Action Block*, bzw. *Activity*) ausgewählt werden. Die *Aktion* wird dann im Zustand abgelegt und näher spezifiziert. Die Angaben *entry:* und *exit:* müssen ebenfalls eingetragen werden.	*Maus: 6* *Tastatur: 3* *9*
04 Anlegen der Subzustände von *Produktwahl*	Vorher muß der Superzustand mittels Drag&Drop vergrößert werden, um die anderen Zustände aufnehmen zu können. Danach DK auf das Werkzeug *State* und Plazieren der Subzustände im Zustand *Produktwahl*, außerdem Eingabe der Bezeichner. Ebenso Anlegen von Start- und Endzustand. **SELECT Enterprise kann keine History-Zustände erzeugen.**	*Maus: 15* *Tastatur: 4* *19*
05 Bearbeiten der Subzustände	Analog Punkt 03	*Maus: 10* *Tastatur: 5* *15*

Schritt	Vorgangsweise	Benötigte Aktionen
06 Anlegen der Zustandsübergänge, die mit dem Zustand *leer* in Verbindung stehen	Auswahl des Werkzeugs *Transition*, Anlegen sämtlicher Zustandsübergänge mit gehaltener STRG-Taste durch LK in den Quell- und dann LK in den Zielzustand. Danach Auswahl des Werkzeugs *Event Action Block* und LK in die Transition, um einen neuen Block anzulegen. Textuelle Beschreibung der Parameter des Zustandsübergangs im zugehörigen Eigenschaftenfenster (nicht direkt im Diagramm möglich).	*Maus:* 11 *Tastatur:* 2 13
07 Anlegen der noch nicht vorhandenen Zustandsübergänge, die mit dem Zustand *Produktwahl* in Verbindung stehen	Analog Punkt 06	*Maus:* 18 *Tastatur:* 3 21
08 Anlegen der Zustandsübergänge, die mit dem Zustand *Auswahl* in Verbindung stehen	Analog Punkt 06, beim Anlegen des rekursiven Zustandsübergangs, muß zweimal in die leere Diagrammfläche geklickt werden.	*Maus:* 25 *Tastatur:* 4 29
09 Anlegen der restlichen Zustandsübergänge	Analog Punkt 06	*Maus:* 23 *Tastatur:* 4 27
Gesamt		***153***

6.5.7 Zustandsdiagramm - StP/UML

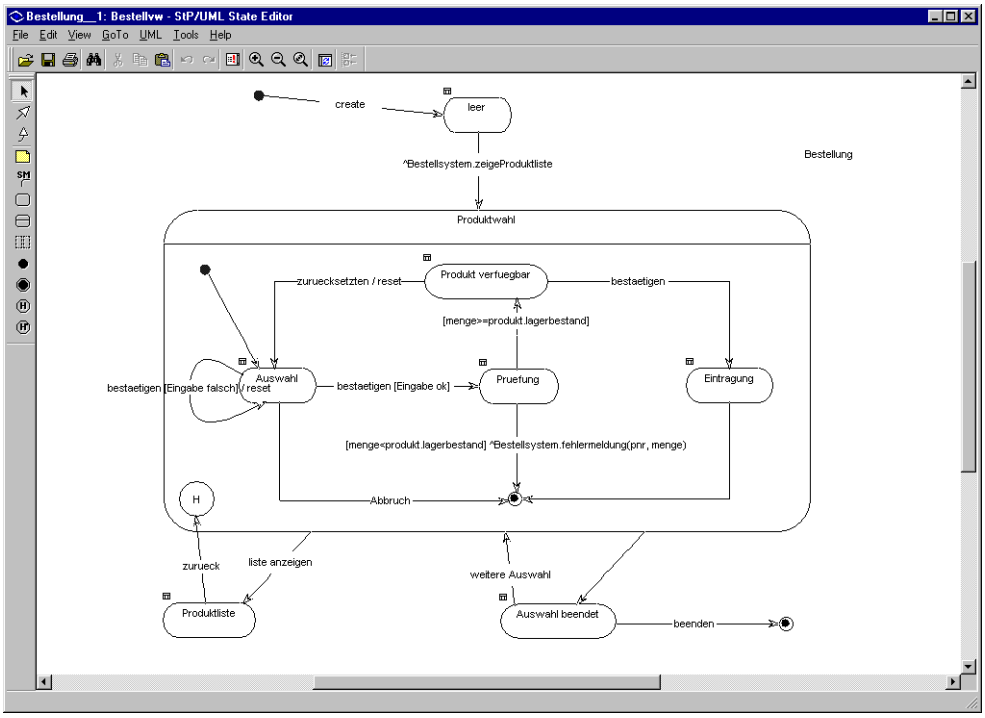

Abb. 6–32
Zustandsdiagramm in StP/UML

Schritt	Vorgangsweise	Benötigte Aktionen
01 Anlegen eines neuen Zustandsdiagramms für die Klasse Bestellung	LK auf die Klasse Bestellung im Klassendiagramm, Auswahl des Menüpunktes *GoTo - State Diagram for Class*. Ein Dialog fragt ab, ob ein neues Zustandsdiagramm erstellt werden soll.	*Maus: 4* *Tastatur: 0* *4*
02 Anlegen von Start-, End-, eines Super-, und drei "normaler" Zustände mit Bezeichnern	Auswahl folgender Werkzeuge und LK in die Diagrammfläche, um den entsprechenden Zustand anzulegen: ❑ *Initial State* ❑ *Final State* ❑ *Composite State* ❑ *State* Um mehrere gleichartige Zustände nacheinander einzutragen, wird das Werkzeug *State* mittels DK ausgewählt. Die Bezeichner der Zustände werden sofort nach Anlegen derselben eingetragen.	*Maus: 10* *Tastatur: 4* *14*

Schritt	Vorgangsweise	Benötigte Aktionen
03 Bearbeiten der Zustände	Um einen Zustand näher zu beschreiben, d.h. interne Aktionen u.a. einzutragen, muß der betreffende Zustand ausgewählt werden und dann über den Menüpunkt *GoTo - State Table* der Table-Editor des Zustands aufgerufen werden. Ist noch keine Tabelle angelegt worden, wir die Erzeugung zuvor noch in einem Dialog abgefragt. Im Editor können nun die Informationen in den entsprechenden Zellen der Tabelle (z.B. *Entry Action*, oder *Activity*) eingetragen werden, wobei die Einhaltung der Syntax bestimmter Aktionen (wie etwa Send-Messages) beim Benutzer selbst liegt. Abschließend muß noch abgespeichert werden, bevor wieder in das Diagramm durch Schließen des Editor-Fensters zurückgekehrt wird. Leider ist es nicht möglich, die eingetragenen Informationen auch im Klassendiagramm anzeigen zu lassen.	*Maus: 15* *Tastatur: 3* *18*
04 Anlegen der Subzustände von *Produktwahl*	Vorher muß der Superzustand mittels Drag&Drop vergrößert werden, um die anderen Zustände aufnehmen zu können. Danach DK auf das Werkzeug *State* und Plazieren der Subzustände im Zustand *Produktwahl*, außerdem Eingabe der Bezeichner. Ebenso Anlegen von Start- und Endzustand, sowie eines History-Zustands (Werkzeug *Shallow History*).	*Maus: 15* *Tastatur: 4* *19*
05 Bearbeiten der Subzustände	Analog Punkt 03	*Maus: 25* *Tastatur: 5* *30*
06 Anlegen der Zustandsübergänge, die mit dem Zustand *leer* in Verbindung stehen	DK auf das Werkzeug *Arc* und Verbinden der Start- mit den Zielzuständen. Nach Anlegen der Kanten folgt der Eintrag der Parameter des Zustandsübergangs: Deselektion des Werkzeugs, RK auf die entsprechende Kante und Bestätigen des Kontextmenüeintrags *Properties*. Eintrag der Parameter als Freitext-Bezeichner im folgenden Dialog, sowie Bestätigung.	*Maus: 11* *Tastatur: 2* *13*
07 Anlegen der noch nicht vorhandenen Zustandsübergänge, die mit dem Zustand *Produktwahl* in Verbindung stehen	Analog Punkt 06	*Maus: 19* *Tastatur: 3* *22*

Schritt	Vorgangsweise	Benötigte Aktionen
08 Anlegen der Zustandsübergänge, die mit dem Zustand *Auswahl* in Verbindung stehen	Analog Punkt 06, beim Anlegen des rekursiven Zustandsübergangs, muß zweimal in die leere Diagrammfläche geklickt werden.	*Maus: 26* *Tastatur: 4* *30*
09 Anlegen der restlichen Zustandsübergänge	Analog Punkt 06	*Maus: 25* *Tastatur: 4* *29*
Gesamt		***179***

6.5.8 Zustandsdiagramm - WithClass

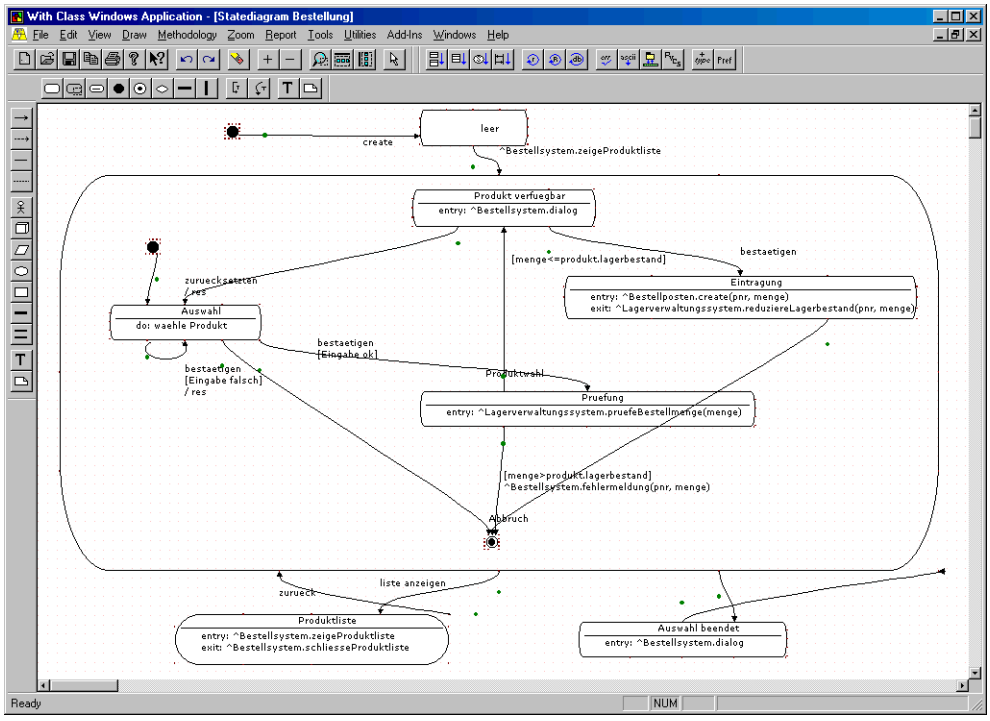

Abb. 6–33
Zustandsdiagramm in WithClass

Schritt	Vorgangsweise	Benötigte Aktionen
01 Anlegen eines neuen Zustandsdiagramms für die Klasse Bestellung	Auswahl des Menüpunktes *View - New Diagram - State*, Eintrag des Bezeichners und Öffnen des Diagramms	*Maus: 3* *Tastatur: 1* *4*

Schritt	Vorgangsweise	Benötigte Aktionen
02 Anlegen der Subzustände von *Produktwahl*	Auswahl des Werkzeugs *Wait State*, LK in die, um einen Zustand anzulegen. Vergabe des Bezeichners direkt nach Anlegen eines Zustands über ein Textfeld. Zusätzlich Anlegen des Start- und des Endzustands mit den Werzeugen *Initial State* und *Final State*. **WithClass kann keine History-Zustände darstellen.**	*Maus: 12* *Tastatur: 4* *16*
03 Bearbeiten der Subzustände	DK auf den entsprechenden Zustand, um den Eigenschaften-Dialog zu öffnen. Folgende Informationen müssen nun spezifiziert werden: ❑ *Do Activity* - Eintrag in ein Textfeld ❑ *Show Do Text* - Aktivierung ❑ *State Class* - Auswahl der zugeordneten Klasse aus einer Dropdown-Liste, da dies nicht anders möglich ist, jedoch für die Konsistenz eines Modells unentbehrlich, wird dieser Vorgang auch bewertet Weiters ist es danach notwendig, die Zustände händisch mittels Drag&Drop zu vergrößern, da sonst die eingetragenen Informationen nur abgeschnitten dargestellt würden.	*Maus: 44* *Tastatur: 4* *48*
04 Anlegen der übrigen Zustände	Analog Punkt 02 Der Superzustand wird modelliert, indem um die Subzustände ein Rahmen gezogen wird, danach erfolgt ein LK auf das Werkzeug *Composite State*. Ein Bezeichner muß noch im folgenden Dialog eingegeben und bestätigt werden. **Achtung:** Da die Testversion von WithClass auf eine maximale Anzahl von 10 möglichen Zuständen limitiert ist, entfällt die Modellierung des Endzustands (bzw. der zugehörigen Kante)	*Maus: 12* *Tastatur: 4* *16*
05 Bearbeiten der Zustände	Analog Punkt 03	*Maus: 32* *Tastatur: 4* *36*
06 Anlegen der Zustandsübergänge, die mit dem Zustand *leer* in Verbindung stehen	Auswahl des Werkzeugs *Transition*, Anlegen des Zustandsübergangs und anschließend Beschreibung der Parameter des Zustandsübergangs im Textfeld.	*Maus: 6* *Tastatur: 2* *8*

Schritt	Vorgangsweise	Benötigte Aktionen
07 Anlegen der noch nicht vorhandenen Zustandsübergänge, die mit dem Zustand *Produktwahl* in Verbindung stehen	Analog Punkt 06	*Maus: 12* *Tastatur: 3* *15*
08 Anlegen der Zustandsübergänge, die mit dem Zustand *Auswahl* in Verbindung stehen	Analog Punkt 06, beim Anlegen des rekursiven Zustandsübergangs, muß nur zweimal den selben Zustand geklickt werden.	*Maus: 12* *Tastatur: 4* *16*
09 Anlegen der restlichen Zustandsübergänge	Analog Punkt 06	*Maus: 15* *Tastatur: 4* *19*
Gesamt		***178***

6.6 Round-Trip Engineering

An dieser Stelle soll die Funktionalität und Performanz der Programme in Bezug auf den Vorgang des *Round-Trip Engineering*, d.h. Erzeugen von Quellcode (*Codegenerierung*) sowie Rückführen der im Code enthaltenen Informationen in das statische Modell (*Reverse-Engineering*) anhand eines einfachen, praktischen Beispiels analysiert werden.

Zuerst wird das Projekt teilweise in den Werkzeugen modelliert und der Code generiert. In der nächsten Phase wird der automatisch erzeugte Code händisch erweitert. Dabei wird eine Methode einer bestehenden Klasse mit Code versehen und darüberhinaus eine neue Klasse eingefügt, die eine Referenz auf eine schon bestehende Klasse enthält.

Nun wird der Code wieder in das Modell übernommen, wobei die neue Klasse richtig erkannt werden, und der händisch erweiterte Code erhalten werden sollte. Da der außerhalb des Werkzeugs hinzugefügte Code nicht bei jedem Programm im Modell aufscheint und so nicht überprüft werden kann, ob dieser erhalten bleiben würde, wird abschließend ein weiteres Mal der Codegenerator ausgelöst.

Eine Beschreibung der Funktionalität erfolgt für jedes Werkzeug getrennt. Um einen empirischen Vergleich der Geschwindigkeit der Funktionen in Abhängigkeit von der Größe des entsprechenden Modells zu ermöglichen, wird für das erwähnte Modell eine Zeitmessung durchgeführt, deren Ergeb-

nisse abschließend mit denen eines etwas komplexeren Modells verglichen werden.

Die einzelnen Phasen des Testdurchlaufs werden an dieser Stelle noch näher beschrieben:

- *Phase 1*
  Folgendes Modell wird in den Tools angelegt, danach der Code vorwärts generiert (siehe Abb. 6–34).

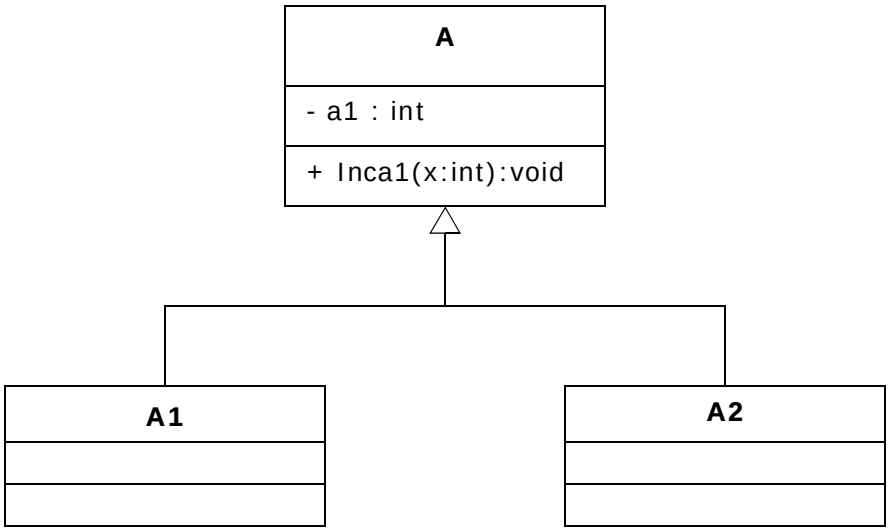

Abb. 6–34
UML-Modell für die Codegenerierung

- *Phase 2*
  Dem generierten Code werden folgende Programmstücke hinzugefügt:

```
A::Inca1(int x)
{
   a1+=x;
}

class B
{
   int b1;
   A *ref;
   void decB1(int x);
};
```

  Der erweiterte Code wird anschließend rückwärts generiert und so in das Modell aufgenommen. Das Tool sollte in der Lage sein, die Informationen korrekt zu interpretieren und sowohl das Modell konsistent zu halten, als auch das Klassendiagramm aktualisiert darzustellen.

- *Phase 3*
  Ausgehend vom aktualisierten Modell wird der Code erneut generiert. Die Fragestellung in diesem Zusammenhang ist, ob die Anweisung in der Prozedur vom Codegenerator überschrieben oder beibehalten wird.

- *Phase 4*
  Abschließend wird die Performanz der Werkzeuge anhand eines weiteren Projekts analysiert: Eine bestehende C++ Header-Datei wird ein-

gelesen und in ein Klassendiagramm rückgeführt. Anschließend erfolgt eine Codegenerierung aus den Informationen des statischen Modells.
Der Umfang des gewählten Modells ist hierbei gegenüber dem vorigen um ein Vielfaches größer und umfaßt:

- 20 Klassen mit jeweils 2 Attributen und einer Operation
- 6 Assoziationen
- 8 Generalisationsbeziehungen

Anm.: Alle dargestellten Codebeispiele sind Auszüge aus dem gesamten generierten Code. Ferner sind die Codeteile eventuell aus verschiedenen Dateien entnommen und leicht nachformatiert, um den Platzbedarf zu reduzieren.

6.6.1 Round-Trip Engineering - Cool:Jex

Zu Beginn wird ein neues Projekt im Repository und darin eine Konfigurationsversion angelegt. In der Konfigurationsversion werden zumindest zwei Phasenversionen benötigt, eine Objektdesign-Phase, um das System zu modellieren, und eine Implementationsphase, in der die Codegenerierung stattfindet. In der Objektdesignphase wird nun ein Paket angelegt, das sämtliche Elemente des Beispielprojektes aufnehmen wird.

Jetzt wird in dem Paket ein Klassendiagramm angelegt, und in diesem das Modell (Phase 1) konstruiert.

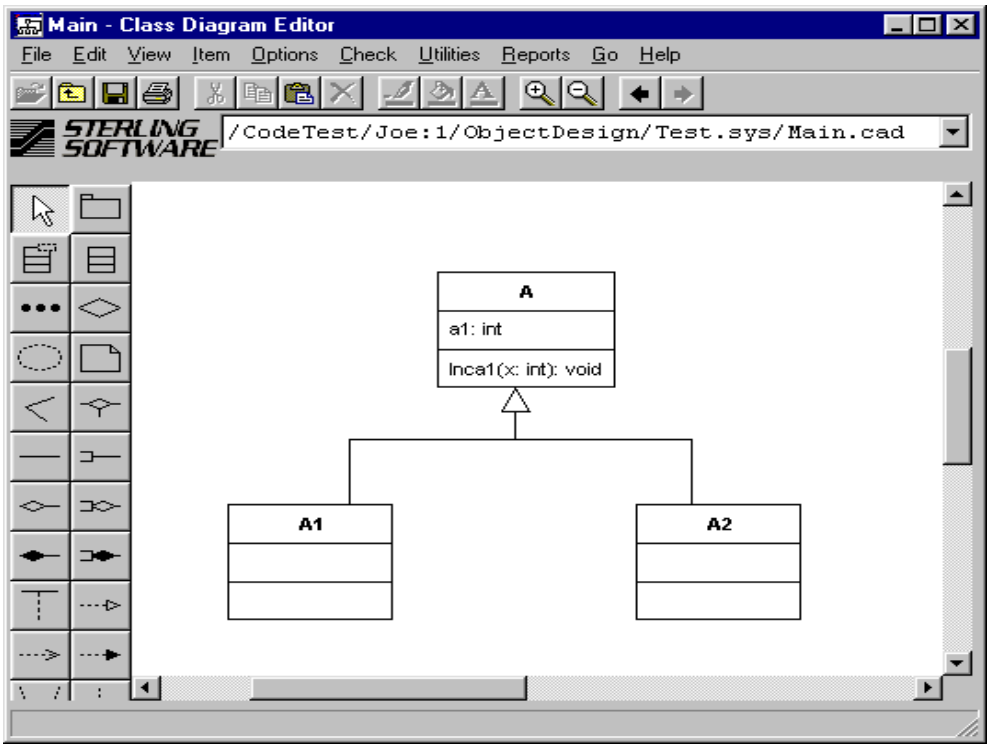

Abb. 6–35
UML-Modell (Phase 1) in Cool:Jex

Vor der Codegenerierung sollte eventuell mit dem Menübefehl *Check Local Model* überprüft werden, ob das System korrekt modelliert wurde, da sämtliche Parameter von Attributen und Methoden textuell in das Diagramm geschrieben werden, und sich so leicht Syntaxfehler einschleichen.

Ist das Modell korrekt, kann der Codegenerator gestartet werden. Dazu wird im Browser der Eintrag für die Implementationsphase markiert, worauf in der Menüleiste ein zusätzliches Menü mit dem Titel *C++* erscheint (auch *Java* usw., wenn das entsprechende Modul aktiv ist, im folgenden wird jedoch ausschließlich auf den Codegenerierungsprozeß für C++ eingegangen). Hier findet sich der Befehl *Generate Specific Packages*, mit dem das vorhandene Paket aus der Objektdesign-Phase in die Implementationsphase übernommen werden kann. Bei diesem Vorgang, dem Importieren eines Paketes in die Implementationsphase, wird für dieses Code generiert. Cool:Jex informiert den Benutzer, wie bei jeder länger andauernden Aktion, in einem Log-Fenster über den aktuellen Fortschritt. Hier werden auch eventuell auftretende Probleme und Fehler ausgegeben.

Code aus Phase 1, generiert von Cool:Jex

```
class A {
public:
   // Default constructor/destructor
   A();
   virtual ~A();
   // User-defined methods
   void Inca1(long x);
   // Attribute accessor methods
   long getA1() const;
   void setA1(long newA1);
   // Association accessor methods
protected:
   // User-defined methods
   // Attribute accessor methods
   // Association accessor methods
private:
   // User-defined methods
   // Attribute accessor methods
   // Association accessor methods
   // User-defined attributes
   long a1;
   // Association attribute storage
};

inline long A::getA1() const
{
   return a1;
}

inline void A::setA1(long newA1)
{
   a1 = newA1;
}

class A1 : public A {
public:
   // Default constructor/destructor
   A1();
   ~A1();
   // User-defined methods
   // Attribute accessor methods
   // Association accessor methods
protected:
```

```
   // User-defined methods
   // Attribute accessor methods
   // Association accessor methods
private:
   // User-defined methods
   // Attribute accessor methods
   // Association accessor methods
   // User-defined attributes
   // Association attribute storage
};

class A2 : public A {
public:
   // Default constructor/destructor
   A2();
   ~A2();
   // User-defined methods
   // Attribute accessor methods
   // Association accessor methods
protected:
   // User-defined methods
   // Attribute accessor methods
   // Association accessor methods
private:
   // User-defined methods
   // Attribute accessor methods
   // Association accessor methods
   // User-defined attributes
   // Association attribute storage
};
```

Das Programm generiert automatisch Konstruktoren und Destruktoren für alle Klassen. Weiters werden Zugriffsfunktionen für alle Attribute einer Klasse erzeugt. Cool:Jex legt für jede Klasse automatisch eine Header- und eine Codedatei an.

Im nächsten Schritt werden in den vom Programm generierten Dateien Änderungen vorgenommen, und diese wieder in das Modell übernommen. Die Methode Inca1() der Klasse A wird mit Code versehen, und eine Klasse B wird eingefügt.

Nachdem die Änderungen an den Codedateien vorgenommen sind, wird das Ergebnis in das Modell rückgeführt. Hierzu muß im Browser das Paket in der Objektdesignphase markiert, und aus dem *C++*-Menü der Befehl *Reverse Engineer* ausgewählt werden. Im folgenden Dialog kann unter anderem festgelegt werden, ob auch Assoziationen generiert werden sollen, und ob vorhandener Code, der über die Deklaration hinausgeht übernommen werden soll (*Code Injection*).

Während des Reverse Engineering öffnet Cool:Jex wieder ein Log-Fenster um den Benutzer zu informieren. Nach kurzer Zeit ist der Vorgang abgeschlossen, und im Browserfenster, als Inhalt des Paketes in der Objektdesign-Phase, werden einige neue Elemente angezeigt. Dies ist die Klasse B, die neu in das Modell aufgenommen wurde und zwei neue Klassendiagramme, B und ATree. Das Diagramm B stellt die Klassen A und B und ihre Beziehung dar, die von Cool:Jex als Aggregation interpretiert wird. ATree enthält die Klassen A, A1 und A2 und deren Vererbungsbeziehung. Das Klassendiagramm, das bereits vorhanden war, wird von Cool:Jex nicht verändert.

Soll nach eventuellen Erweiterungen im Modell der Code erneut erstellt werden, müssen die zu generierenden Dateien im Browser markiert werden. Dazu wird das Paket in der Implementationsphase markiert, worauf in der rechten Fensterhälfte dessen Inhalt angezeigt wird. Hier werden die gewünschten Dateien markiert, wobei selbstverständlich eine Mehrfachauswahl möglich ist, und über den Menübefehl *C++ - Generate - Selected*, die Codegenerierung erneut durchgeführt werden kann.

Der händisch ergänzte Code der Funktion Inca1(), der im Modell nicht aufscheint, bleibt bei der Neugenerierung des Codes erhalten.

6.6.2 Round-Trip Engineering - objectiF

Zuerst wird in einem neuen System ein Paket angelegt und in diesem ein Klassendiagramm eingerichtet. In diesem Diagramm wird das System (Phase 1) konstruiert.

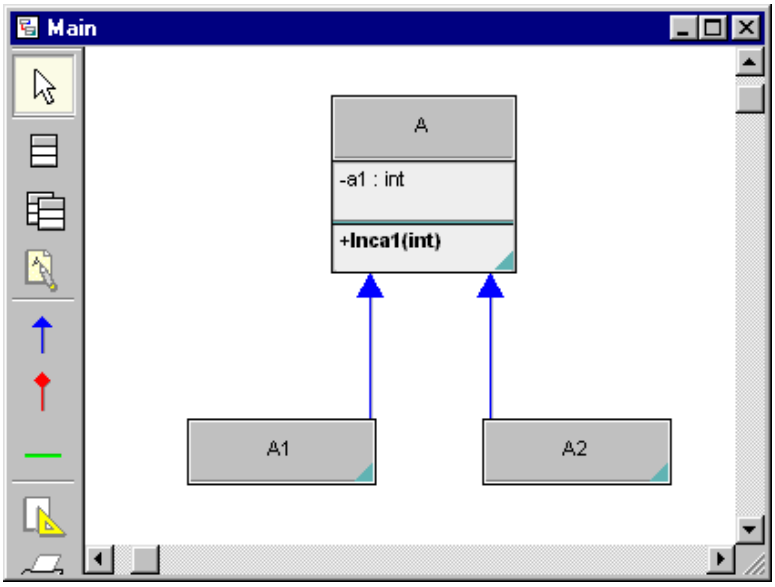

Abb. 6–36
UML-Modell (Phase 1) in objectiF

Jetzt wird dem System der C++ Skript-Server zugewiesen. Dies geschieht durch einen Rechtsklick auf eine freie Stelle im Paket-Browser und Ausführen des Kontextmenübefehls *Skripte*. Mit dem Befehl *Sprache zuweisen* aus dem gleichen Menü wird dem System nun die C++ Sprachkomponente zugewiesen.

Im nächsten Schritt werden im Paket zwei Dateien angelegt, eine Header- und eine Codedatei. Den Dateien müssen nun jeweils alle drei Klassen des Projektes zugeordnet werden.

Sind alle zuvor beschriebenen Aktionen erledigt, kann mittels Kontextmenübefehl (*Generieren*) für die jeweilige Datei, deren Code generiert werden. Als zusätzliches Feature bietet objectiF die Möglichkeit, den Code mit dem Befehl *Code* in einem Editorfenster nur anzusehen. Das Programm informiert den Benutzer mittels Sanduhr über den laufenden Codegenerierungsvorgang. Nach dessen Abschluß wird in einem Nachrichtenfenster bekanntgegeben, ob die Aktion erfolgreich beendet wurde oder nicht.

Code aus Phase 1, generiert von objectiF

```
class CA
{
```

```
   //..begin "UserDefined"
   // put some additional code here
   //..end "UserDefined"
      private:
      int m_a1;
   public:
      void Inca1 (int x);
};

class CA1 :
   public CA
{
   //..begin "UserDefined"
   // put some additional code here
   //..end "UserDefined"
};

class CA2 :
      public CA
{
      //..begin "UserDefined"
      // put some additional code here
      //..end "UserDefined"
};

void CA::Inca1(int x)
{
   // Put your code here
}
```

Optional können für die vorhandenen Klassen automatisch Konstruktor und Destruktor erstellt werden. Die hierfür zuständigen Befehle finden sich im Kontextmenü der jeweiligen Klasse. Der Code den objectiF generiert ist gut strukturiert und nicht mit Kommentaren überladen. Alle Funktionen und auch die Klassendeklarationen besitzen einen mit Kommentaren gekennzeichneten Bereich, in dem der Benutzer händisch Code einfügen kann.

Jetzt werden in den Codedateien manuell Änderungen vorgenommen, und die geänderten Dateien wieder in das Modell zurückgeführt. Hierzu wird die Methode Inca1() der Klasse A mit Code erweitert, und außerdem die Klasse B eingefügt.

Der Inhalt der Dateien wird in das System übertragen indem entweder der Befehl *Dateien rückführen*, oder *Reverse Engineering* benutzt wird. Das zuerst genannte Kommando aktualisiert Modellinformationen, die bereits im System vorhanden sind. Für die *.cpp Datei, in der die händisch erweiterte Funktion Inca1() definiert ist, bedeutet das, daß das Programm den zusätzlichen Code registriert und ins System aufnimmt. Die veränderte Headerdatei muß mit dem Befehl *Reverse Engineering* in das System aufgenommen werden, da hier eine neue Klasse im System angelegt werden muß. Beide Befehle sind sowohl über das Kontextmenü der Datei, als auch über das des Paketes zu erreichen.

Bei der Rückführung werden alle Informationen richtig erkannt und in das Repository übernommen. Will man den geänderten Sachverhalt auch im Klassendiagramm darstellen, muß die neue Klasse und die Assoziation zur Klasse A manuell übertragen werden. Die Elemente sind bereits im Datenstamm vorhanden und müssen nur mehr in das Diagramm übernommen werden. Die Klasse wird mit dem Werkzeug *Klasse übernehmen* in das Dia-

gramm eingezeichnet, die Assoziation muß neu angelegt werden. Beim Anlegen der Assoziation muß zusätzlich angegeben werden, daß diese sich auf das bereits vorhandene Attribut *ref* bezieht, da objectiF sonst bei erneuter Codegenerierung ein weiteres Attribut für die Beziehung erstellt.

Der manuell eingefügte Code kann, anders als bei den meisten anderen Werkzeugen, auch innerhalb des Programms betrachtet und verändert werden. Selbstverständlich bleibt dieser bei einer neuerlichen Codegenerierung erhalten.

6.6.3 Round-Trip Engineering - OTW-2

Es wird ein neues Paket im Paketbrowser erstellt und darin ein Klassendiagramm angelegt. In diesem Diagramm werden die ersten drei Klassen (Phase 1) des Beispielprojektes angelegt. Das Klassendiagramm präsentiert sich wie folgt:

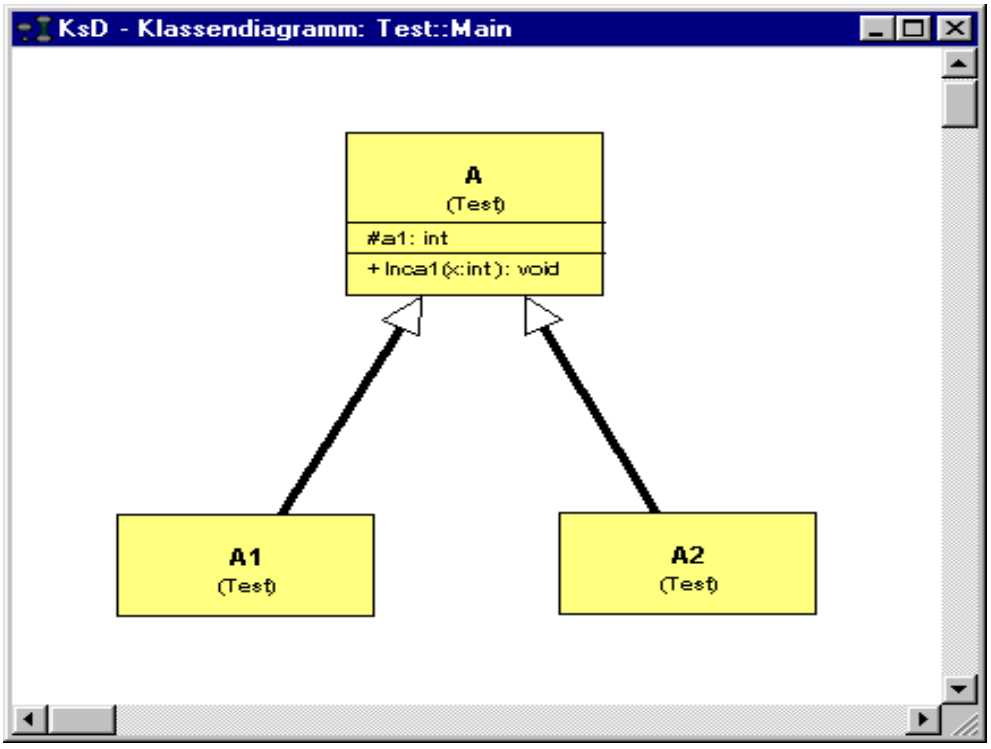

Abb. 6–37
UML-Modell (Phase 1) in OTW-2

Nun werden die Klassen einem Modul zugeordnet. Dies geschieht durch Anklicken der Klassen im Browser oder im Diagramm mit der rechten Maustaste und Auswahl des Kontextmenübefehls *Modul zuordnen*. Danach muß im Eigenschaftsdialog des Moduls die Schaltfläche *Scannen/Generieren* aktiviert werden, damit der Codegenerator bzw. Scanner das Modul bearbeiten kann. Hier können auch Dateinamen für die zu erzeugenden Dateien festgelegt werden.

Jetzt kann der Codegenerator gestartet werden. Dies geschieht durch Auswahl des Kontextmenübefehls *C++ Modulcode vorwärts generieren* aus dem Kontextmenü des Moduls. Das Werkzeug generiert eine Header- und eine Quellcodedatei. Die Klassendeklarationen sehen folgendermaßen:

Code aus Phase 1, generiert von OTW-2 (ohne Standardfunktionen)

```
//********** Forward- und Enumerationdeklarationen **********
```

```
class A;
class A1;
class A2;

//************************** class A **************************
class A {
public:
   voidInca1(int);
protected:
   inta1;
};

//************************** class A1 **************************
class A1 : public A {
};

//************************** class A2 **************************
class A2 : public A {
};
```

Optional können für die vorhandenen Klassen automatisch Standardfunktionen wie Konstruktor, Destruktor und Vergleichsoperatoren generiert werden. Dazu existiert im Eigenschaftsdialog der jeweiligen Klasse, eine Schaltfläche *Standardfunktionen generieren*.

Code aus Phase 1, generiert von OTW-2 (mit Standardfunktionen)

```
//************* Forward- und Enumerationdeklarationen **************
class A;
class A1;
class A2;

//************************ class A **************************
class A {
public:
   voidInca1(int);
   A();
   A(const A&);
   virtual ~A();
   A& operator=(const A&);
   booloperator==(const A&) const;
protected:
   inta1;
};

//************************** class A1 **************************
class A1 : public A {
public:
   A1();
   A1(const A1&);
   virtual ~A1();
   A1&operator=(const A1&);
   booloperator==(const A1&) const;
};

//************************** class A2 **************************
class A2 : public A {
public:
```

```
   A2();
   A2(const A2&);
   virtual ~A2();
   A2&operator=(const A2&);
   booloperator==(const A2&) const;
};
```

Jetzt werden in den vom Programm generierten Codedateien händisch Änderungen vorgenommen, und das Ergebnis wieder in das Modell zurückgeführt. Hierzu wird die Methode Inca1() der Klasse A mit Code versehen, und zusätzlich die Klasse B eingefügt, die mit A assoziiert ist.

Nachdem die Änderungen an den Codedateien vorgenommen sind wird der Scanner von OTW-2 gestartet, der den Code untersucht, und die gewonnenen Informationen in das Modell übernimmt. Dies geschieht über das Kontextmenü des Moduls über den Befehl *C++ Modul scannen*.

OTW-2 meldet den laufenden Scanvorgang über ein Statusfenster an den Benutzer und informiert, welche Zeile gerade bearbeitet wird. Nach wenigen Augenblicken ist der Code in das Modell übernommen. Alle Informationen werden richtig eingelesen, und in das Repository übernommen. Will man den geänderten Sachverhalt auch im Klassendiagramm darstellen, muß man die neue Klasse und die Assoziation zur Klasse A händisch einzeichnen. Beide Elemente sind bereits im Repository vorhanden und müssen nur mehr in das Diagramm übernommen werden.

Die Codezeile, die der Funktion Inca1() zugefügt wurde, ist im Modell nicht ersichtlich. Sie bleibt dennoch beim nächsten Codegenerierungsvorgang erhalten, wenn der Benutzer den Befehl *C++ Modulcode inkrementell generieren* wählt. Wählt man wie zuvor *C++ Modulcode vorwärts generieren*, wird der Code auf Basis des Modells neu erstellt, und dadurch alle händischen Erweiterungen zerstört.

6.6.4 Round-Trip Engineering - Paradigm Plus

Zuerst wird ein neues Projekt angelegt. Im Klassendiagramm wird ein Paket Test angelegt. Anschließend werden die Klassen zusammen mit Attribut, Operation und Generalisierungshierarchie angelegt.

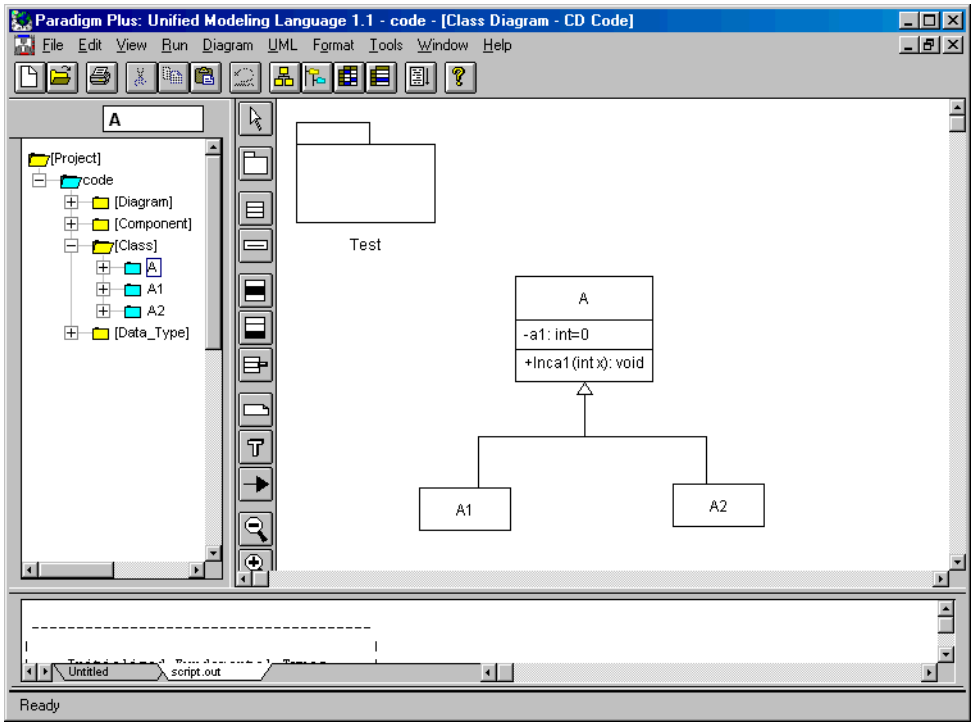

Abb. 6–38
UML-Modell (Phase 1) in Paradigm Plus

Nun wird die Generierung der C++ Coderahmen durchgeführt. Hierzu wird im Menü *Run* der Punkt *Code Generator* ausgewählt. Ein Dialogfenster fragt nun ab, ob die Einstellungen in Hinsicht auf Zielsprache, etc., die beim Anlegen des Projekts definiert wurden korrekt sind (in diesem Fall Generierung von ANSI C++ Code). Nach der Bestätigung muß noch die Klasse angegeben werden, für die Code erzeugt werden soll. Leider kann nicht das Paket ausgewählt werden, sodaß für jede Klasse separat Code erzeugt wird.

Paradigm Plus erstellt für jede Klasse ein Header-, sowie ein Implementationsfile. Es werden alle Definitionen aus dem Modell übernommen, zusätzlich werden nur Konstruktor / Destruktor für die Klasse generiert. Der Code ist sehr übersichtlich, auf Kommentare, welche die Lesbarkeit beeinträchtigen könnten, wird positiverweise verzichtet.

Code aus Phase 1, generiert von Paradigm Plus

```
#ifndef A_H
#define A_H

//##begin class A
//+========================================================================+
//| CLASS: A
//|-------------------------------------------------------------------------|
//| ATTRIBUTES:
//|    int a1
//|-------------------------------------------------------------------------|
//| OPERATIONS:
//|    virtual void Inca1(int x)
//|-------------------------------------------------------------------------|
//| Who | Date       | Description
//|-------------------------------------------------------------------------|
//|  P+ | Dec 16, 99 | Last Code Generation
//+========================================================================+

class A {
```

```
   private:
     int a1;
   public:
     A(void);
     ~A(void);
     virtual void Inca1(int x);
};
//##end class A

#endif
```

Nun wird den erstellten Dateien händisch Code hinzugefügt. Dabei wird die Methode Inca1() im File A.cpp mit Code versehen und in A.h eine neue Klasse B erstellt, die eine Assoziation mit A besitzt, die über einen Pointer auf dieselbe implementiert wird.

Der geänderte Code wird nun in das Modell übernommen. Dies wird über die Funktion *Import* durchgeführt. In einem Dialogfenster werden zuerst der Typ (ANSI C++) und dann die Bezeichner der Dateien spezifiziert, die eingelesen werden sollen. In diesem Beispiel ist das die geänderte A.h Datei, die nun analysiert und deren Daten in das Repository übernommen werden. Da die Klasse A bereits in der Datenbasis enthalten ist, erfolgt eine sog. Kollisionswarnung, mit der Möglichkeit, die bestehenden Daten zu überschreiben, die unterschiedlichen Informationen zu vereinen, etc. Nachdem die Daten im Repository eingetragen wurden, muß das Diagramm noch aktualisiert werden. Hierfür steht die Funktion *Populate* zur Verfügung, die in mehreren Dialogen den Benutzer zum gewünschten Endergebnis führt.

Wird das Modell nochmals vorwärts generiert, so wird der im Implementationsfile hinzugefügte Code nicht überschrieben.

6.6.5 Round-Trip Engineering - Rational Rose

Die drei Klassen der Phase 1 des Testbeispiels werden in ein leeres Klassendiagramm übertragen und die Vererbungsbeziehung zwischen ihnen angelegt. Anstatt die Klassen einzeln in jeweils zwei Dateien pro Klasse zu generieren, wird eine neue Komponente angelegt, in ein Komponentendiagramm eingetragen, und die drei Klassen dieser Komponente zugewiesen.

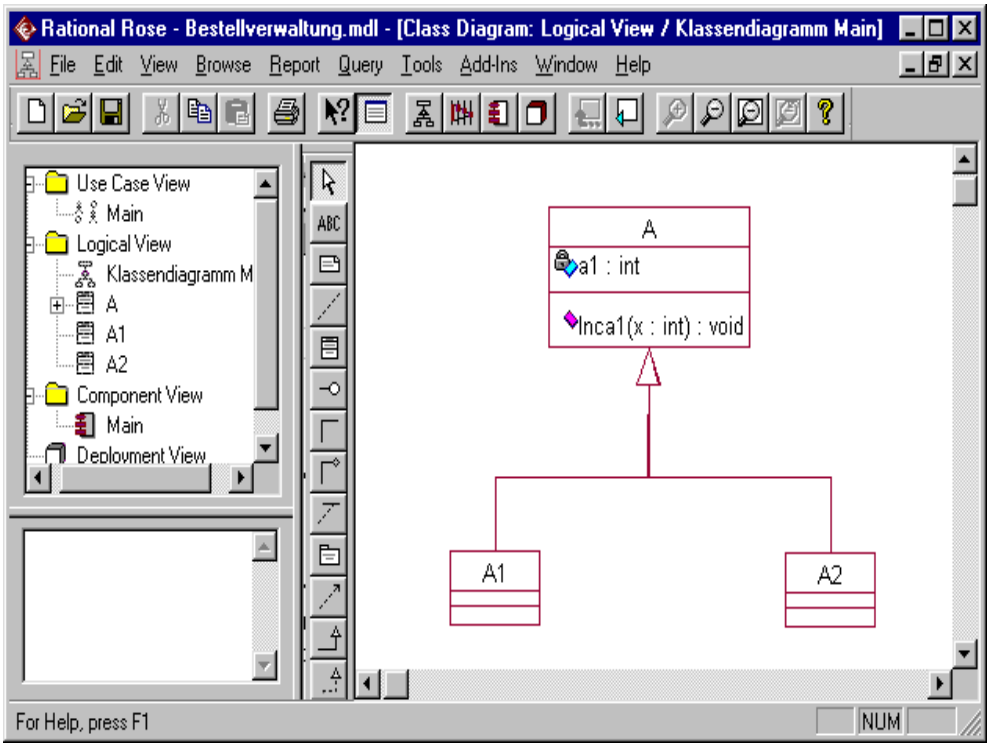

Abb. 6–39
UML-Modell (Phase 1) in Rational Rose

Jetzt kann der Codegenerator gestartet werden. Dies geschieht durch Auswahl des Kontextmenübefehls *Generate Code* aus dem Kontextmenü der Komponente im Diagramm. Das Werkzeug generiert eine Headerdatei, die auch Funktionsdefinitionen enthält. Der erstellte Code ist sehr umfangreich, enthält eine sehr große Anzahl an Konstruktoren und Operatoren und ist mit vielen kryptischen Kommentaren überfrachtet, die nicht als Information für den Benutzer gedacht sind, sondern gespeicherte Information für Rational Rose darstellen. Im folgenden ist nur die Klassendeklaration der Klasse A abgedruckt, da alleine die übrigen Deklarationen den Rahmen gesprengt hätten.

Code aus Phase 1, generiert von Rational Rose

```
class A
{
  //## begin A%3846F11E03DE.initialDeclarations preserve=yes
  //## end A%3846F11E03DE.initialDeclarations

  public:
    //## Constructors (generated)
      A();
      A(const A &right);
    //## Destructor (generated)
      ~A();
    //## Assignment Operation (generated)
      const A & operator=(const A &right);

    //## Equality Operations (generated)
      int operator==(const A &right) const;
      int operator!=(const A &right) const;
    //## Other Operations (specified)
      //## Operation: Inca1%944173599
      void Inca1 (int x);
    // Additional Public Declarations
```

```
      //## begin A%3846F11E03DE.public preserve=yes
      //## end A%3846F11E03DE.public

  protected:
    // Additional Protected Declarations
      //## begin A%3846F11E03DE.protected preserve=yes
      //## end A%3846F11E03DE.protected

  private:
    //## Get and Set Operations for Class Attributes (generated)
      //## Attribute: a1%3846F1AB006E
      const int get_a1 () const;
      void set_a1 (int value);
    // Additional Private Declarations
      //## begin A%3846F11E03DE.private preserve=yes
      //## end A%3846F11E03DE.private

  private: //## implementation
    // Data Members for Class Attributes
      //## begin A::a1%3846F1AB006E.attr preserve=no  private: int {U}
      int a1;
      //## end A::a1%3846F1AB006E.attr
    // Additional Implementation Declarations
      //## begin A%3846F11E03DE.implementation preserve=yes
      //## end A%3846F11E03DE.implementation
};

//## begin A%3846F11E03DE.postscript preserve=yes
//## end A%3846F11E03DE.postscript
//## begin A1%3846F1230262.preface preserve=yes
//## end A1%3846F1230262.preface
//## Class: A1%3846F1230262
//## Category: Sam%3846F3850230
//## Subsystem: <Top Level>
//## Persistence: Transient
//## Cardinality/Multiplicity: n
```

Als Nächstes wird den vom Programm generierten Dateien händisch Code hinzugefügt, und das Ergebnis wieder in das Modell zurückgeführt. Dabei wird die Methode Inca1() mit Code versehen, und die Klasse B in das System eingefügt. In dem von Rational Rose generierten Code befinden sich innerhalb jedes Funktionsrumpfes zwei Kommentarzeilen, die Anfang und Ende des sogenannten *Body*-Bereiches definieren. Extern erweiterter Code muß zwischen diesen beiden Kommentaren eingefügt werden, wenn er vom Tool erhalten werden soll.

Rational Rose bietet zum Zweck des Reverse Engineering ein Zusatzwerkzeug. In diesem *C++ Analyzer* muß zuerst ein leeres Projekt angelegt werden. In dem Projekt werden nun die Codedateien spezifiziert, die analysiert werden sollen. Weiters können verschiedene zusätzliche Pfade für die Dateiein- und Ausgabe sowie die Suche nach Include-Dateien, und Basisprojekte angegeben werden. Das sind Projekte, die zumindest teilweise im aktuellen Projekt benötigt werden (z.B. Klassenbibliotheken).

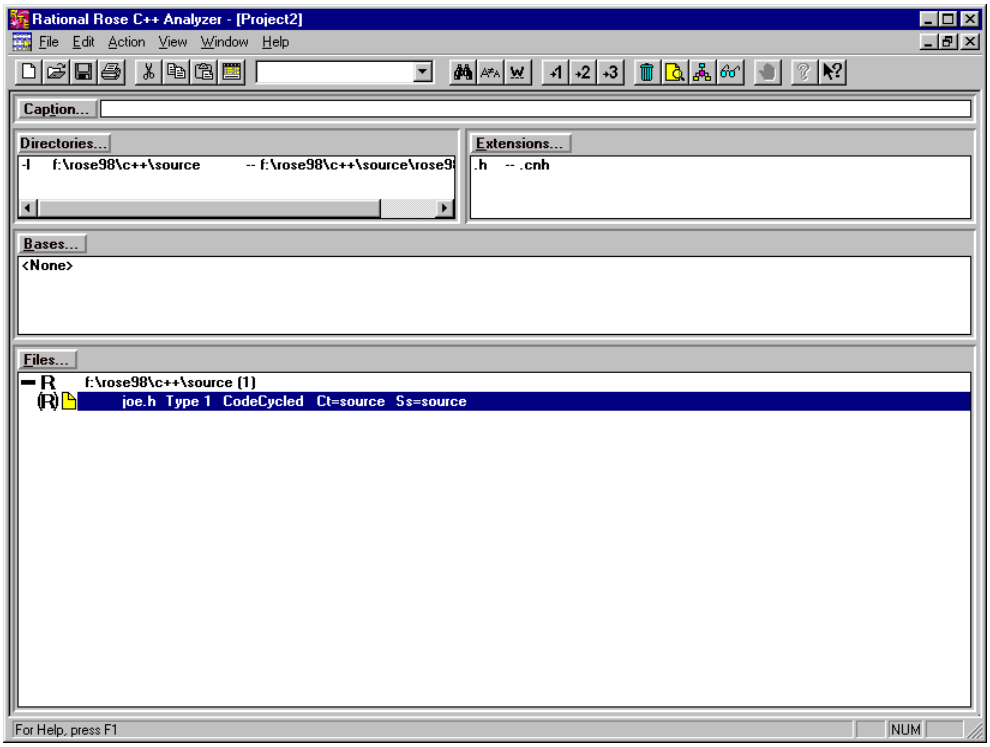

Abb. 6–40
Der Rational Rose C++ Analyzer

Sind alle Parameter festgelegt, wird der Analysevorgang gestartet, und der Benutzer erhält Informationen über die Analyse in einem Log-Fenster. War die Analyse erfolgreich, kann das Ergebnis exportiert und von Rational Rose genutzt werden. Einerseits kann die Information als *.mdl-Datei, und somit als vollständig neues Projekt gespeichert werden. Ist es jedoch, wie im vorliegenden Fall, erwünscht die Information in ein bereits bestehendes Modell einzufügen, kann eine *.red-Datei angelegt werden, die mit dem Befehl *Update* in ein bestehendes Rose-Modell importiert werden kann.

Nach dem Import der *.red-Datei in das zuvor angefertigte Modell, existiert im System ein neues Paket namens Source, das die Klasse B enthält. Alle Informationen aus der Code-Datei sind im System vorhanden, jedoch kann die Assoziation nicht graphisch dargestellt werden. Sie ist vorhanden und kann über die Eigenschaftsdialoge der beteiligten Klassen erreicht werden, trotzdem ist es nicht möglich, sie in das Klassendiagramm einzutragen.

Wird der Code ein weiteres Mal generiert, bleibt das manuell eingefügte Codestück der Methode Inca1() erhalten.

6.6.6 Round-Trip Engineering - SELECT Enterprise

Zuerst wird ein neues Projekt angelegt. Im Klassendiagramm wird ein Paket Test angelegt. Anschließend werden die Klassen zusammen mit Attribut, Operation und Generalisierungshierarchie im Paket angelegt.

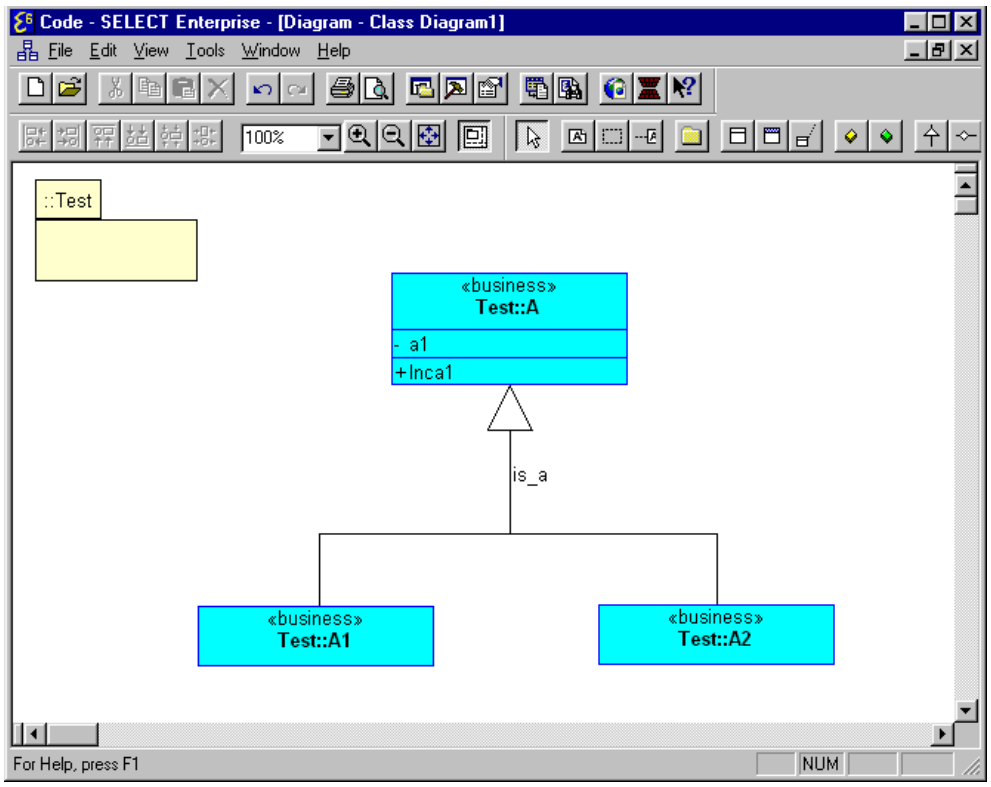

Abb. 6–41
UML-Modell (Phase 1) in SELECT Enterprise

Die statische Klassenstruktur soll nun in C++ Code umgesetzt werden. Hierzu wird über das Menü *Tools* der *SELECT Enterprise C++ Generator* aufgerufen. Im folgenden Dialogfenster erhält der Benutzer eine Übersicht über die zu verarbeitenden Klassen, kann jedoch auch auf Wunsch einen weiteren Dialog öffnen, in dem eine sehr große Anzahl an allgemeinen Einstellungen spezifiziert werden können. Ein weiterer Dialog bietet darüber hinaus noch die Möglichkeiten von separaten Einstellungen für die einzelnen Klassen (z.B. ob Zugriffsfunktionen erzeugt werden sollen, die Art der Implementierung von Attributen und Operationen, u.v.m.). Die Codegenerierung wird nun ausgeführt.

SELECT Enterprise erstellt für jede Klasse ein Header-, sowie ein Implementationsfile. Alle Klassen eines Pakets werden in einem eigenen Verzeichnis gespeichert.

Datei A.h:

Code aus Phase 1, generiert von SELECT Enterprise

```
#ifndef __A_H
#define __A_H

//{{SCG_HEADER(A.h) [0]
//{{SCG_INCLUDE
//}}SCG_INCLUDE
//{{SCG_FORWARD
//}}SCG_FORWARD
//{{SCG_CLASS(0)
//{{SCG_CLASS_INFO(0)

class A

//}}SCG_CLASS_INFO
```

```
{
    //{{SCG_CLASS_PROPS(0)
private:
    int a1;
public:
    void Inca1(int x = 0);
    //}}SCG_CLASS_PROPS
};
//}}SCG_CLASS
//}}SCG_HEADER
#endif
```

Datei A.cpp:

```
//{{SCG_IMPLEMENTATION(A.cpp) [0]
//{{SCG_INCLUDE
#include "A.h"
//}}SCG_INCLUDE

//{{SCG_ICLASS_INFO(0)
//}}SCG_ICLASS_INFO
//{{SCG_ATTS(0)
//}}SCG_ATTS
//{{SCG_ROLES(0)
//}}SCG_ROLES
//{{SCG_OPS(0)
//{{SCG_OP(0.0)
//{{SCG_OP_INFO
//////////////////////////////////////////////////////////////////////

void A::Inca1(int x )
//}}SCG_OP_INFO
{
  // Code to be added here
}
//}}SCG_OP
//}}SCG_OPS
//}}SCG_IMPLEMENTATION
```

Nun wird den erstellten Dateien händisch Code hinzugefügt. Dabei wird die Methode Inca1() im File A.cpp mit Code versehen, und zwar anstatt der Markierung `//code to be added here`. In A.h wird eine neue Klasse B erstellt, die eine Assoziation mit A besitzt, die über einen Pointer auf dieselbe implementiert wird.

Der geänderte Code wird nun in das Modell übernommen. Hierzu wird der *Synchronizer* gestartet, der nach einem Analysevorgang sehr übersichtlich die Unterschiede zwischen dem Modell und den im Code enthaltenen Informationen darstellt:

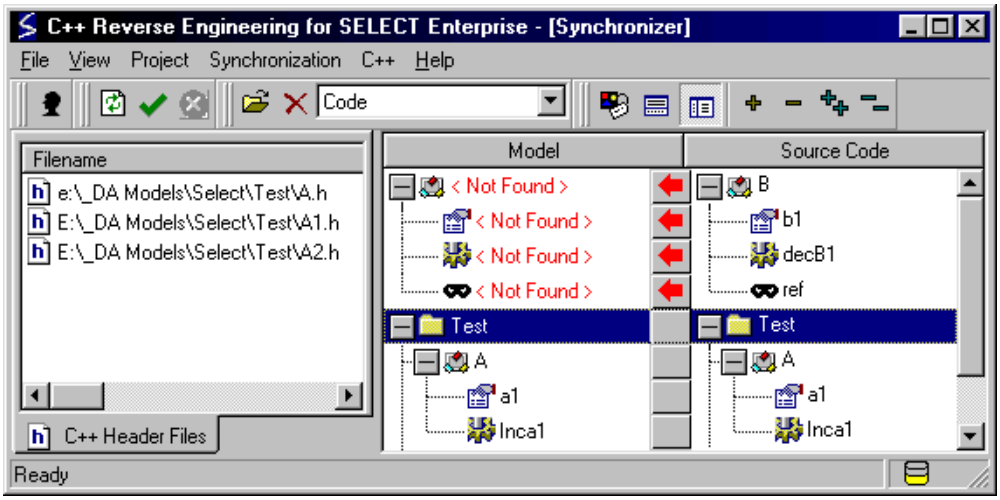

Abb. 6–42
Der SELECT C++ Synchronizer

Das Aktualisieren des Modells wird durch Auswahl des Befehls *Commit* durchgeführt. Im Klassendiagramm wird die neue Klasse B über den Befehl *Populate* eingezeichnet. Leider wird die Assoziation nicht korrekt erkannt und interpretiert. Bei einer erneuten Vorwärtsgenerierung wird der hinzugefügte Code nicht verändert, auch die Klassendeklaration von B bleibt bestehen, SELECT Enterprise erstellt einfach eine zweite im selben File.

6.6.7 Round-Trip Engineering - StP/UML

Zuerst wird ein neues Projekt angelegt. Im Klassendiagramm wird ein Paket Test angelegt. Anschließend werden die Klassen zusammen mit Attribut, Operation und Generalisierungshierarchie im Paket angelegt.

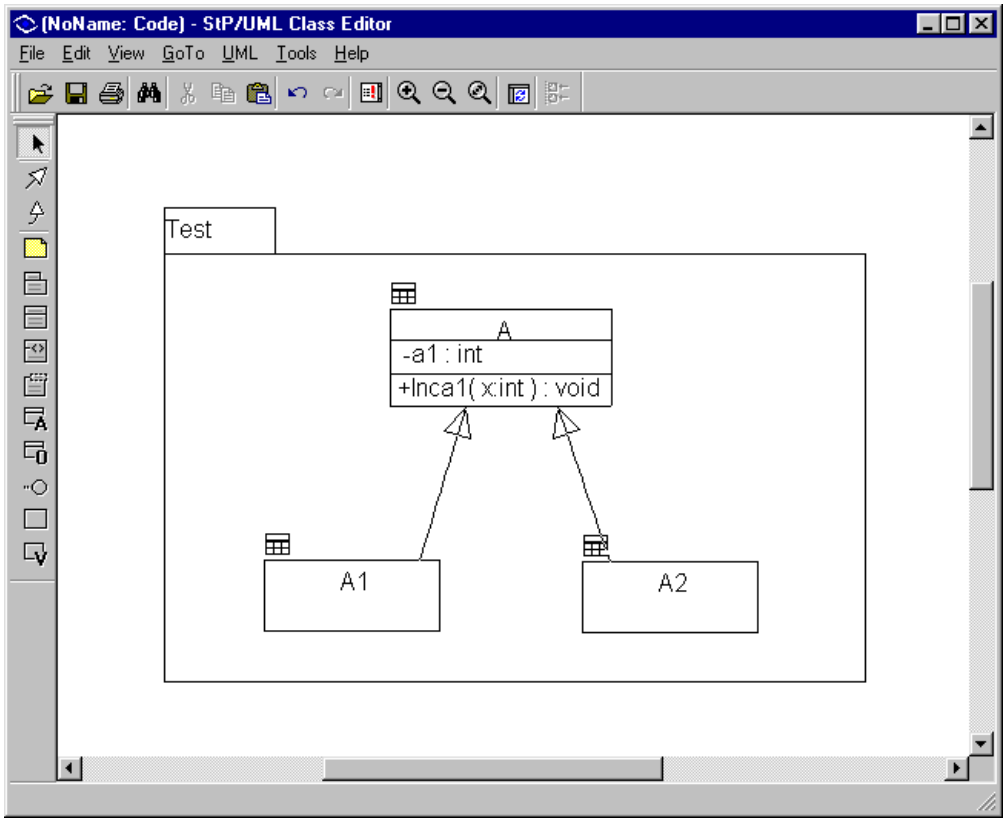

Abb. 6–43
UML-Modell (Phase 1) in StP/UML

Nun wird die Generierung der C++ Coderahmen durchgeführt. Hierzu wird im StP/UML-Desktop im Kontextmenü des Klassendiagramms der Punkt *Generate C++ for Diagram's Classes...* ausgewählt. In einem Dialogfenster können nun noch weitere Angaben gemacht werden, etwa ob ein einzelnes File für alle Klassen erstellt werden soll oder nicht. Es wird jedoch die Defaulteinstellung übernommen, daß für jede Klasse eine separate Zieldatei erzeugt werden soll.

StP/UML erstellt für jede Klasse ein Header-, sowie ein Implementationsfile. Es werden lediglich die Definitionen aus dem Modell übernommen, zusätzliche Details, wie Konstruktor / Destruktor werden nicht generiert. hierfür müßte das entsprechende Script geändert werden.

Das Tool verwendet Kommentare als Codemarkierungen zur Unterscheidung zwischen Codeteilen, die außerhalb, bzw. in StP/UML selbst deklariert wurden (siehe hierzu die Markierungen in der A.cpp Datei - `//stp code` und `//end stp code`).

Datei A.h:

Code aus Phase 1, generiert von StP/UML

```
// StP
// Created on 06/17/99 00:03:14 for Administrator@GODFATHER
//   from system E:/StP/Examples/Code

#ifndef _A_h_
#define _A_h_

// stp class declarations
namespace Test {
    class A;
}
```

```
// end stp class declarations

// stp class definition 0
class Test::A
{
// stp class members
public:
    void Inca1(int x);
protected:
private:
    int a1;
// end stp class members
};
// end stp class definition

// stp footer
#endif
// end stp footer
```

Datei A.cpp:

```
#include "A.h"
// stp operation 0::1
void
Test::A::Inca1(int x)
{

    // stp code

    // end stp code

}
// end stp operation
```

Nun wird den erstellten Dateien händisch Code hinzugefügt. Dabei wird die Methode Inca1() im File A.cpp mit Code versehen, und zwar außerhalb der erwähnten Codemarkierungen, da StP/UML diese Teile bei einer erneuten Vorwärtsgenerierung überschreiben würde. In A.h wird eine neue Klasse B erstellt, die eine Assoziation mit A besitzt, die über einen Pointer auf dieselbe implementiert wird.

Der geänderte Code wird nun in das Modell übernommen. Dies wird über die Funktion *Parse Source Code...* im Menü *Code - Reverse Engineering* durchgeführt. In einem Dialogfenster werden die Dateien angegeben, die analysiert werden sollen.

Nach Abschluß der Operation wird der Befehl *Generate Model from Parsed Source Files..* ausgeführt und die Daten werden in das Repository übernommen. Außerdem erstellt StP/UML ein Klassendiagramm, das die beiden Klassen A und B enthält, sowie deren Assoziation. Weiters kann die Diagrammdarstellung basierend auf den Informationen in der Datenbasis aktualisiert werden.

6.6.8 Round-Trip Engineering - WithClass

Zuerst wird ein neues Projekt angelegt. Im Klassendiagramm wird ein Paket Test angelegt. Dann wird die Klassenstruktur angelegt und abschließend werden über den Eigenschaften-Dialog des Pakets die Klassen diesem hinzugefügt.

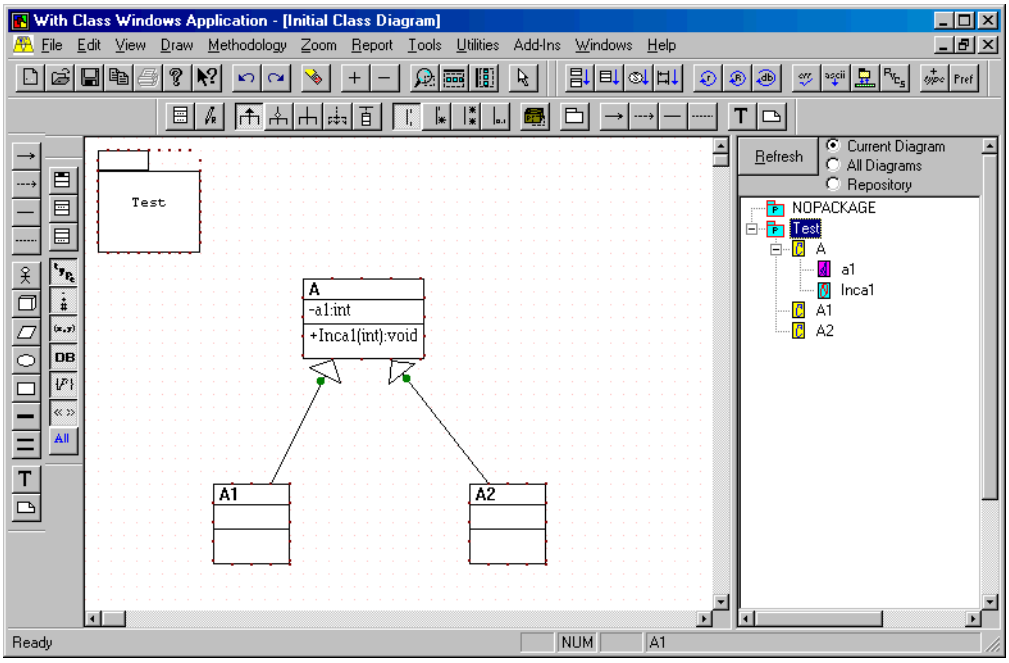

Abb. 6–44
UML-Modell (Phase 1) in WithClass

Nun wird die Generierung der C++ Coderahmen durchgeführt. Hierzu wird im Menü *Utilities* der Punkt *Generate Code File per Class...* ausgewählt. Im folgenden Dialogfenster können verschiedene fertige Scripts ausgewählt werden, die jeweils für eine bestimmte Zielsprache den entsprechenden Sourcecode erzeugen. Es wird das Script *C++ Class Header* ausgewählt. Außerdem wird das Paket Test als Quelle der Klassen bestimmt. Nach der Bestätigung wird der Vorgang ausgeführt.

WithClass erstellt für jede Klasse ein eigenes Header-File, andere Einstellungen müßten über die Scripts definiert werden. Der Code wirkt ein wenig durch Kommentare überfrachtet, die Hinweise für den Benutzer darstellen, ist aber sonst übersichtlich und komplett. Es werden alle notwendigen Funktionen implementiert, wie Konstruktor, Destruktor, sowie verschiedene Zugriffsmethoden. Auch die Generalisation wird korrekt abgebildet, der Übersicht wegen wird hier jedoch nur der Code der Klasse A angeführt.

Code aus Phase 1, generiert von WithClass

```
// Class: A                        //ANSI C++

#ifndef __A_H  //Required for current class
  #define __A_H

#ifndef __IOSTREAM_H              //Required for cin and cout
  #include <iostream.h>
#endif

class A
{
   int a1; //Attribute data member
```

```
protected:

public:
           //Default constructor alternative to compiler provided
default constructor
           //Ensure correct initial values
           //Initialization list has members in the order declared
           //Association object data member pointers initialized to null
association object
  A ()    : a1(0)
  {
           //Initialization of array of 1:M association objects to null
association objects
  }
           //Constructor with arguments
           //Update to argument list to initialize base class data
members,
           //e.g. (int aNumber) : BaseClass (aNumber)
  A (    int aa1  )
  {
     a1 = aa1;
           //Initialization of array of 1:M association objects to null
association objects
  }
   //Copy constructor alternative to compiler provided default copy
constructor
   //Copy alternatives for association objects: (1) initialize
association object to nullAssociation Object
   //(2) Shallow copy to copy pointers of association objects (3) Deep
copy to create new association objects
   //and copy values of association objects
   //Commented out code assigns 1:1 and 1:M association object data
member pointers for shallow copy
   //Remove // if you desire to assign pointers
  A (const A& aA )
  { int i = 0;
    a1 = aA.a1;
  }
   //Operator= Assignment Operator alternative to compiler provided
default operator=
   //If base class make virtual
   //Assignment alternatives for association objects: (1) initialize
association object to nullAssociation Object
   //(2) Shallow copy to copy pointers to association objects (3) Deep
copy to create new association objects
   //and copy values of association objects
   //Commented out code assigns 1:1 association object data member
pointers for shallow copy
   //Remove // if you desire to assign association object pointers for
shallow copy
  A& operator= (const A& aA);

           //Operator== Equality Operator - No compiler default
operator== generated
```

```
          //If base class make virtual
          //Update to access 1:1 association object data members
          //Function does not compare 1:1 and 1:M association object
data member pointers
  int operator== (const A& aA) const;

          //Operator<< for cout
  friend ostream& operator<< (ostream& os, A& aA);

          //Operator>> for cin
  friend istream& operator>> (istream& is, A& aA);

                      //Get accessor function for non-static attribute
data member
  int geta1() const
  { return a1;
  }

                      //Set accessor function for non-static attribute
data member
  void seta1 (const int aa1)
  { a1 = aa1;
  }

   void Inca1 (int x=0)    ;

   ~ A ( ) { }  //Destructor - Delete any pointer data members that used
new in constructors
    //Destructor should be virtual if and only if class contains at least
one virtual function
    //Objects destroyed in the reverse order ot the construction order
};

#endif
```

Nun wird den erstellten Dateien händisch Code hinzugefügt. Dabei wird die Methode Inca1() im File A.cpp mit Code versehen und in A.h eine neue Klasse B erstellt, die eine Assoziation mit A besitzt, die über einen Pointer auf dieselbe implementiert wird.

Der geänderte Code wird nun in das Modell übernommen. Dazu wird im Menü *Utilities* der Punkt *Reverse File...* ausgewählt. Die entsprechende Datei wird angegeben und der Vorgang angestoßen. WithClass analysiert die Datei und aktualisiert sofort das Klassendiagramm. Es werden sämtliche Informationen korrekt dargestellt. Der Code der Methode kann gesondert eingelesen werden, hierzu wird im Eigenschaftendialog der Operation die Funktion *Import C++*, oder *Update* angewählt. Das hinzugefügte Codestück wird ebenfalls ohne Probleme eingelesen und kann jetzt im Tool weiter verändert werden. Wird der Import nicht durchgeführt, so wird der Code bei einem erneuten Anstoßen der Vorwärtsgenerierung überschreiben.

6.6.9 Ergebnisse der empirischen Performanz-Messung

In Abb. 6–45 und Abb. 6–46 werden die Ergebnisse der Zeitmessung dargestellt, jeweils für das erste Modell (*Test A*) und das zweite, größere Modell (*Test B*). Die Tests wurden gesamt auf dem selben Computersystem (P2-350, 128 Mb RAM, Windows-NT 4.0 SP-6) durchgeführt. Die Werte sind auf ganze Sekunden gerundet, sodaß bei mehreren sehr kleinen Werten ein Unterschied eventuell im Bereich von Sekundenbruchteilen liegen kann. Außerdem ist es bei einigen Werkzeugen notwendig, innerhalb des Reverse-Engineerings die Vorgänge des Code-Einlesens (*Parsing*) und des Erstellens des Klassendiagramms getrennt aufzurufen. Hierbei wurden die einzelnen Meßwerte addiert.

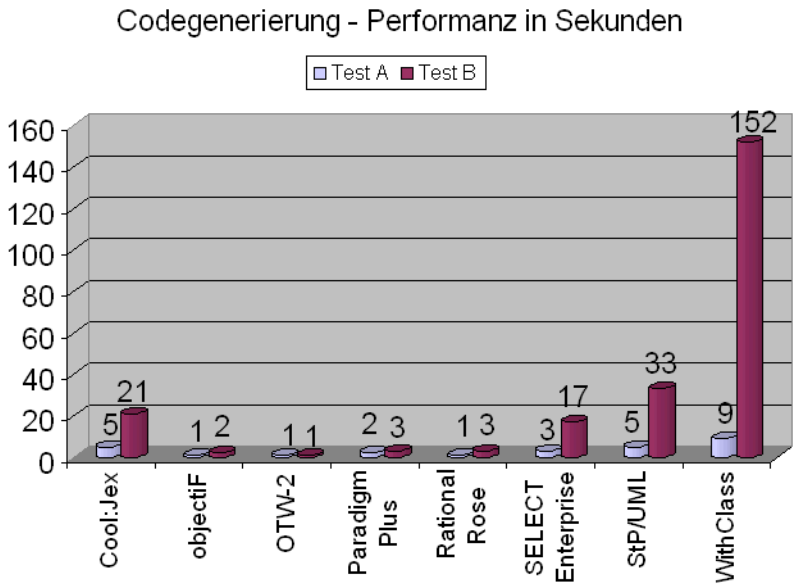

Abb. 6–45
Ergebnisse der Zeitmessung der Codegenerierung

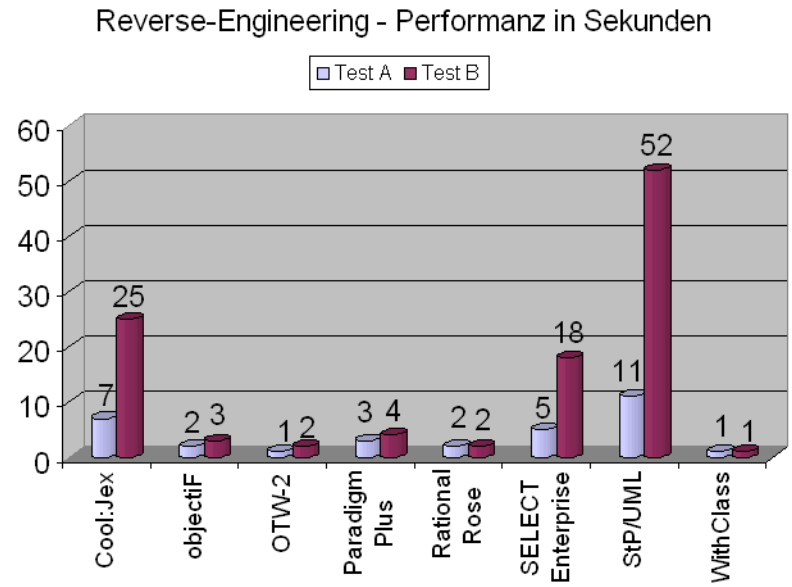

Abb. 6–46
Ergebnisse der Zeitmessung des Reverse-Engineerings

Es ist zu erkennen, daß teilweise große Unterschiede in der Performanz zwischen den einzelnen Werkzeugen bestehen. Beim Vergleich der Ergebnisse

von *Test A* (Modell mit 3 Klassen) bzw. *Test B* (Modell mit 20 Klassen) fällt auf, daß bei manchen Werkzeugen (z.B. Cool:Jex) eine unterproportionale Steigerung des Zeitverbrauchs vorhanden ist. Dies kann auf einen gewissen fixen Overhead (Starten und Initialisieren der benötigten Programmmodule etc.) zurückzuführen sein.

Es kann auch gezeigt werden, daß die sehr umfangreichen Werkzeuge wie erwartet eine vergleichsweise niedrige Performanz aufweisen. Gründe hierfür können unter anderem in der komplexen Datenhaltung liegen. Eine Ausnahme hierbei stellt einerseits Paradigm Plus dar, das zwar über ein entsprechendes Repository verfügt, aber dennoch sehr gute Werte bei der Zeitmessung liefert, auf der anderen Seite ist die Geschwindigkeit der Codegenerierung bei WithClass - in Anbetracht der Größe des Werkzeugs - sehr unbefriedigend.

A Literaturverzeichnis

[Brac95] R. Brachman, Z. Schmolze: An Overview on the KL-ONE Knowledge Representation System. Cognitive Science 9/ 1985, Sep. 1985

[Burk94] R. Burkhardt: Modellierung dynamischer Aspekte mit dem Objekt-Prozeß-Modell. TU Illmenau, Dissertation, 1994

[Burk97] R. Burkhardt: UML - Unified Modeling Language. Addison-Wesley, 1997

[Harm98] P. Harmon, B. Sawyer: UML for Visual Basic 6.0 Developers: Using Visual Modeler and Rational Rose 98. Morgan Kaufmann Publishers, 1998

[Hitz99] M. Hitz, G. Kappel: UML@Work. dpunkt.verlag, 1999

[Oest97] B. Oesterreich: Objektorientierte Softwareentwicklung mit der Unified Modeling Language (3. Auflage). Oldenbourg, 1997

[OMG99] Object Management Group: OMG Unified Modeling Language Specification (draft) V1.3. June 1999, http://www.omg.org

[Quat97] T. Quatrani: Visual Modeling With Rational Rose and UML. Addison-Wesley, 1997

[Rumb99] J. Rumbaugh, I. Jacobson, G. Booch: The Unified Modeling Language Reference Manual. Addison-Wesley, 1999